EXHIBIT 10

U.S. Patent Number 8,352,730 –Infringement Charts

Assignee:	Proxense, LLC
Title:	Biometric personal data key (PDK) authentication
Filing Date:	2005-12-20
Publication Date:	2013-01-08
Inventor:	Giobbi, John J.

'730 Patent Claim[1]		**Accused Instrumentality And Where Each Claim Element Is Found[2]** **Authenticator Implementation**
1	A method comprising[3]:	The preamble is not limiting. Microsoft uses, distributes, sells, and/or offers to sell a Universal Passwordless Architecture ("UPA") which includes: (a) the Microsoft Authenticator app and/or Windows Hello[4] (PROX_MSFT_003067, Plan a passwordless authentication deployment in Azure Active Directory - Microsoft Entra | Microsoft Learn), (b) Microsoft Compatible FIDO2 Security Keys (PROX_MSFT_002844, Become a Microsoft-Compatible FIDO2 Security Key Vendor for sign-in to Azure AD - Microsoft Entra | Microsoft Learn) and/or Windows 10/11 which includes the Cloud Authentication Provider (CloudAP) and Web Account Manager (WAM), (c) Edge and Chrome browser support (PROX_MSFT_002819, All about FIDO2,

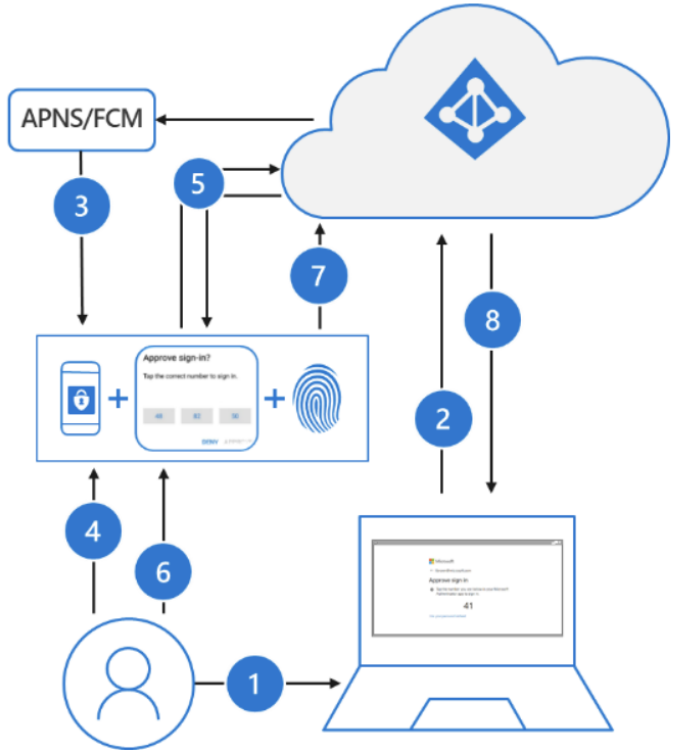
[1] All charts set forth herein for any independent patent claims are hereby incorporated by reference into the charts alleged for any dependent patent claims that depend on such independent claims, as if fully set forth therein.

[2] The Accused Instrumentalities and associated exhibits discussed and/or cited for any claim herein are representative in all material respects of all other accused instrumentalities identified for that claim (e.g., a specified device or service may be used as a representative example in these charts since the other accused instrumentalities have immaterial differences in their hardware and/or software configuration, the cited references are believed to be illustrative of all such accused devices).

[3] Plaintiff's inclusion of any claim preamble in this claim chart should not be interpreted as an admission that the preamble is limiting. Plaintiff reserves the right to take the position that the claim preambles are limiting or not limiting on a claim-by-claim basis. The detail provided in the preamble section is a description of who performs, sells, or offers for sale the method overall and not an admission that the preamble is limiting.

[4] Windows Hello is an alternative to the Microsoft Authenticator app.

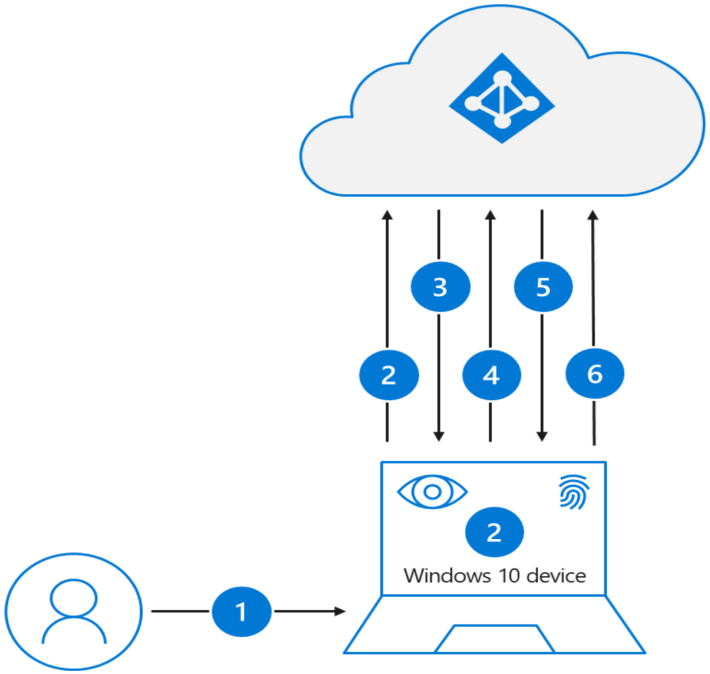
		CTAP2 and WebAuthn - Microsoft Community Hub), and (d) Microsoft Identity (a.k.a., Azure Active Directory and Microsoft Entra) (PROX_MSFT_003066, Passwordless is here and at scale - Microsoft Community Hub), which together infringe, literally or under the doctrine of equivalents, the claimed limitations as presented below. The interoperation of the above elements of the UPA and their use to provide passwordless authentication utilizing the Microsoft Authenticator App on Android and iOS device is detailed in Microsoft's authentication documentation:



1. The user enters their username.
2. Azure AD detects that the user has a strong credential and starts the Strong Credential flow.
3. A notification is sent to the app via Apple Push Notification Service (APNS) on iOS devices, or via Firebase Cloud Messaging (FCM) on Android devices.
4. The user receives the push notification and opens the app.
5. The app calls Azure AD and receives a proof-of-presence challenge and nonce.
6. The user completes the challenge by entering their biometric or PIN to unlock private key.
7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

The interoperation of the above elements of the UPA and their use to provide passwordless authentication utilizing the Windows Hello is detailed in Microsoft's authentication documentation, as reflected below:



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.
6. The Cloud AP provider receives the encrypted PRT with session key. The Cloud AP provider uses the device's private transport key to decrypt the session key and protects the session key using the device's Trusted Platform Module (TPM).
7. The Cloud AP provider returns a successful authentication response to Windows. The user is then able to access Windows as well as cloud and on-premises applications without the need to authenticate again (SSO).

See id.

Microsoft, through its own actions, and the action of its partners, customers, and/or account holders, which Microsoft directs and controls, has created and sells the use of the UPA, including certifying devices and distributing servers and software supporting and integrating into the UPA architecture. Servers integrating with and supporting the architecture include Microsoft Identity servers providing FIDO and/or OAuth services. Devices integrating with and support the architecture include Microsoft Compatible FIDO2 security keys. Software integrating with and supporting the architecture includes

		the Microsoft Edge web browser, support for the Chrome web browser, and the Microsoft Authenticator app and/or Windows 10/11, with Windows Hello.
	persistently storing biometric data of the user and a plurality of codes and other data values comprising a device ID code uniquely identifying the integrated device and a secret decryption value in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered; wherein the biometric data is selected from a group consisting of a palm print, a retinal scan, an iris scan, a hand geometry, a facial recognition, a signature recognition and a voice recognition;	Persistently storing biometric data of a user … in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered Persistent storage of biometric data of a use in a tamper proof format written to a storage element is accomplished via the Microsoft Authenticator app in the UPA. Microsoft's UPA enables account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub ("You can now delete your password from your Microsoft account—or set up a new account with no password—and sign in using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys."). Microsoft's Authenticator app, part of Microsoft's UPA, "turns any iOS or Android phone into a strong, passwordless credential." PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. As shown in the below excerpt from Microsoft's depiction of its UPA, use of the Microsoft Authenticator app on integrated devices, such as Android and iOS devices, to authenticate without passwords requires completing a challenge by scanning biometrics (i.e., biometric verification).

6. The user completes the challenge by entering their biometric or PIN to unlock private key.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

Microsoft, accordingly, designed the Authenticator app to take advantage of biometric features on modern iOS and Android devices and the related functionality in iOS and Android devices. A native feature common to both Android and iOS devices (and utilized by Microsoft's Authenticator app) is the tamper proof storage of biometric information.

Android devices with the Microsoft Authenticator app, for example, persistently store biometric data of the user in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered via a trusted execution environment (TEE). To protect biometric data, Android's implementation guidelines require tamper-proof "raw fingerprint data or derivatives (for example, templates) must never be accessible from outside the sensor driver or TEE" and "fingerprint acquisition, enrollment, and recognition must occur inside the TEE". Android Open-Source Project: Fingerprint HIDL, PROX_MSFT_003026, https://source.android.com/docs/security/features/authentication/fingerprint-hal Accordingly, fingerprint data never leaves the TEE (where both acquisition and recognition occurs). Android's TEE, called Trusty, "uses ARM's Trustzone™ to virtualize the main processor and create a secure trusted exaction environment" isolated from the rest of the system. Android Open-Source Project: Trusty TEE, PROX_MSFT_003088, https://source.android.com/docs/security/features/trusty. Accordingly, fingerprint data, which never leaves the TEE, also never leaves the Trustzone housing Trusty. Keeping biometric data within the Trustzone, Android devices persistently store biometric data in a tamper proof

format.[5] Utilizing Android's persistent storage of biometric information is necessary to complete the challenge by scanning biometrics, as the scanned biometrics must be compared against stored biometric data to determine whether the entered data matches the stored biometric data. Accordingly, by requiring the user complete a biometric "challenge" on an Android device, the Microsoft Authenticator app requires account holders with Android devices to utilize their Android device's TrustZone to persistently store biometric data in a tamper proof format unable to be subsequently altered.

Beginning with Windows 11, Microsoft required that all PCs running Windows 11 must have a TPM to run the operating system with certain features. Intel's latest microprocessors provide this functionality. Intel describes on its website that:

> Trusted platform module (TPM) technology helps keep PCs secure by offering hardware-level protection against malware and sophisticated cyberattacks. TPM technology can be embedded into modern CPUs and "securely store[s] artifacts used to authenticate the platform." The artifacts TPMs protect range from passwords to certificates to fingerprints—any important information users want securely stored.

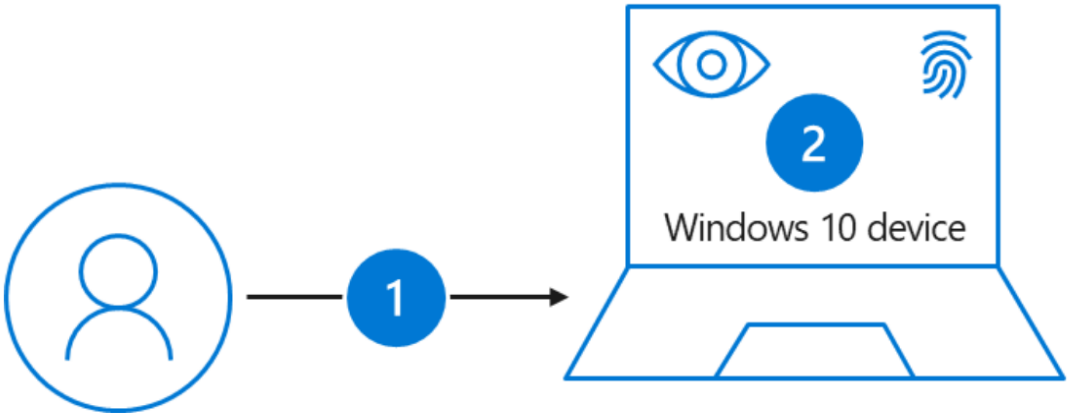
https://www.intel.com/content/www/us/en/business/enterprise-computers/resources/trusted-platform-module.html.

By requiring account holders complete a challenge by scanning biometrics with their Android devices, the Microsoft Authenticator app further requires account holders with Android devices to utilize Fingerprint Hardware Interface Definition Language (HIDL). On Android devices, access to biometric hardware is controlled by Fingerprint HIDL. "Android uses Fingerprint Hardware Interface Definition Language (HIDL) to connect to a vendor-specific library and fingerprint hardware (for example, a fingerprint sensor)." See Android Open Source Project: Fingerprint HIDL, PROX_MSFT_003026, https://source.android.com/docs/security/features/authentication/fingerprint-hal. The methods enabled by the Fingerprint HIDL do not permit altering biometric data. *Id*. (providing a listing of methods, none of which allowing for the alternation of biometric data). Keeping fingerprint and other biometric data within a portion of the Trustzone only accessible by Fingerprint HIDL, which lacks a method for altering biometric data, Android devices persistently store biometric

[5] In the alternative, to the extent the TEE and Trustzone are considered to be a different security implementation or if limited alterations are possible in limited circumstances and given high levels of security permissions, clearance, or hardware-level permission attribution, the "tamper proof format . . . that is unable to be subsequently altered," limitation is met under the doctrine of equivalents on the grounds that the tamper proofing and limitations on alteration utilized in the Trustzone prevent nearly all alteration by nearly all programs that have not been explicitly approved and, at least, prevent alteration by all regular-world (non-TEE and non-Trustzone) applications and requires a strict level of physical and/or virtual layers of permission to be altered.

		data of the user in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered. Again, by requiring account holders with Android devices enter biometrics with Fingerprint HIDL, the Microsoft Authenticator app forces account holder with Android device to persistently store biometric data of the user in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered. In addition to Android devices, the Microsoft Authenticator app is available for use by account holders with iOS devices. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn ("The Authenticator App turns any iOS or Android phone into a strong, passwordless credential."). Accordingly, accounts holders with iOS device must complete a biometric challenger by scanning biometrics into their iOS device. By having the user complete a challenge by scanning their biometric on an iOS device, the Microsoft Authenticator app requires account holders with iOS devices to utilize their integrated device's native biometric features. Once such feature is the persistent storage of biometric information in a tamper proof formation unable to be subsequently altered. Utilizing the persistent storage of biometric information available in iOS devices will be necessary to complete the challenge by scanning biometrics, as the scanned biometrics must be compared against stored biometric data to determine whether the entered data matches the stored biometric data. When the Authenticator App is installed on an iOS device, biometric data will be stored within Apple's Secure Enclave. "The Secure Enclave is a dedicated secure subsystem … isolated from the main processor to provide an extra layer of security designed to keep sensitive data secure even when the Application Processor kernel becomes compromised." Apple Platform Security, page 9. Protecting data even when a hack or malware comprises the Application Processor, the Secure Enclave provides a tamper proof format for sensitive data. The sensitive data protected by Apple's Secure Enclave includes biometric data. "During enrollment, the Secure Enclave processes, encrypts, and stores the corresponding Touch ID and Face ID template data." Apple Platform Security, page 19. Accordingly, by requiring account holders with iOS devices enter biometrics to complete a challenge, the Microsoft Authenticator App also requires that account holders with iOS device utilize the device's secure enclave to persistently store biometric data of the user in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered. Additionally, devices running Microsoft's authenticator app (e.g., iOS and Android devices) require user consent to enroll a fingerprint. *See* PROX_MSFT_003092, Use Touch ID on iPhone and iPad - Apple Support (Detailing the process of adding a fingerprint to an iPhone as including entering the user's passcode prior to adding a fingerprint.); and

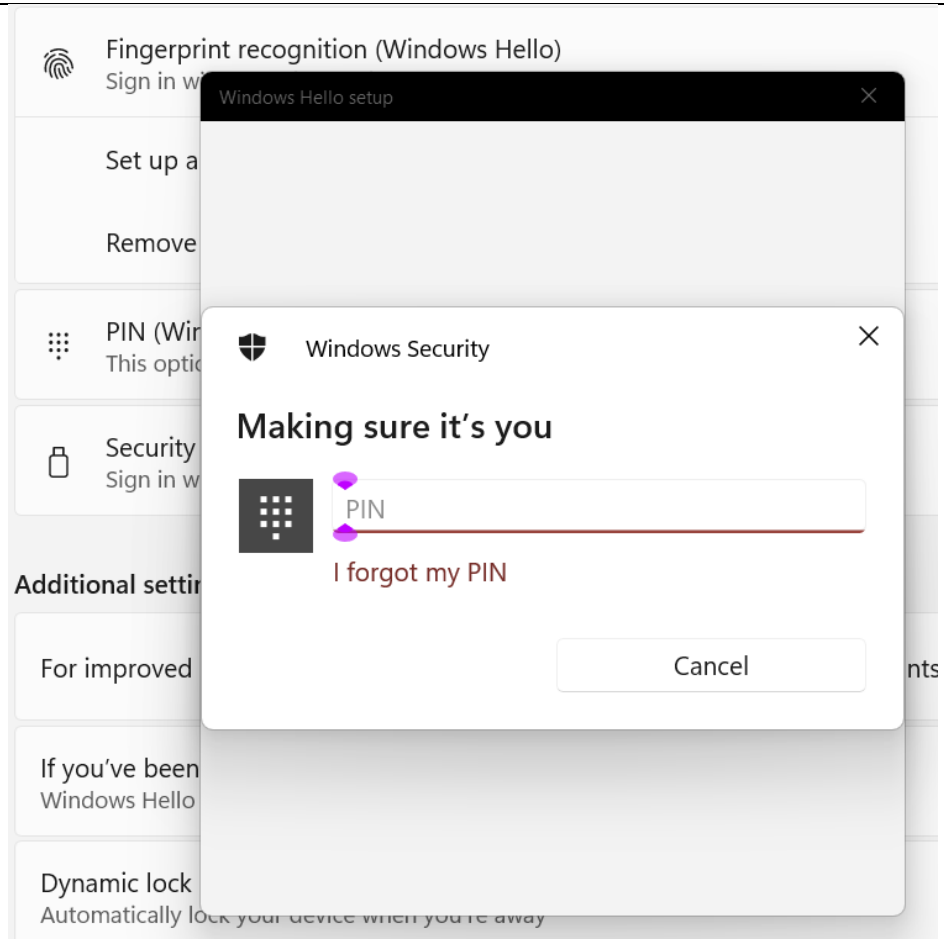
		PROX_MSFT_003031, How to Add a New Fingerprint to Your Android Device - Technipages (Detailing the process of adding a fingerprint to an Android phones as including entering a PIN or password for the device before adding a fingerprint.). As enrolling fingerprints on both iOS and Android devices require entering PIN / Passcode / Password to evidence user consent, iOS and Android devices store biometric data of users in a tamper proof format unable to be subsequently altered. In requiring the user to complete a "challenge" by scanning their biometrics on either an iOS or Android device, the Microsoft Authenticator app requires account holders to enroll a fingerprint with use consent. Enrolling and persistently storing biometric information will be necessary to complete the challenge by scanning biometrics, as the scanned biometrics must be compared against enrolled and stored biometric data to determine whether the entered data matches the stored biometric data. For at least the foregoing, the Microsoft Authenticator App requires account holders with either Android and iOS devices store biometric data of a tamper proof format that is not capable of being subsequently altered (i.e., in order to complete a challenge by scanning biometrics in order to utilize Microsoft's UPA). As Microsoft's Authenticator app "turns any iOS or Android phone into a strong, passwordless credential" Microsoft exercises control and direction over account holders by controlling every step that the software performs and every prompt for user action, to store their biometric data in a tamper proof format written to a storage element on the integrated device that is not capable of being subsequently altered. Additionally, turning to Windows Hello, persistent storage of biometric data of a use in a tamper proof format written to a storage element is accomplished via Windows 10/11. Microsoft's UPA enables account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub ("You can now delete your password from your Microsoft account—or set up a new account with no password—and sign in using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys."). As shown below, use of Windows Hello on integrated devices, such as personal computer and laptops, to authenticate without passwords requires a user signs into window using a biometric gesture (i.e., biometric verification).



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

Persistent storage of biometric information is necessary to sign in a user with a biometric gesture, as the scanned biometrics must be compared against stored biometric data to determine whether the scanned biometric data from a user matches the stored biometric data. Windows 10/11 devices store biometrics in one of two places: 1) a biometric database file for each sensor; or 2) sensor module itself. PROX_MSFT_003297, Windows Hello biometrics in the enterprise - Windows Security | Microsoft Learn ("Each sensor on a device will have its own biometric database file where template data is stored… Some fingerprint sensors have the capability to complete matching on the fingerprint sensor module instead of in the OS. These sensors will store biometric data on the fingerprint module instead of in the database file."). Accordingly, Windows Hello requires persistent storage of biometric data.



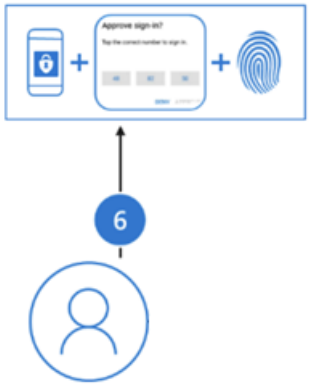
Additionally, as shown in the above screenshot captured in Windows 11, Windows 10/11 requires user consent to enroll a fingerprint. Enrolling fingerprints require entering a PIN to evidence user consent; accordingly, Windows 10/11 devices store biometric data of users in a tamper proof format unable to be subsequently altered.

Furthermore, Microsoft states that "if an attacker was actually able to get the biometric data from a device, it cannot be converted back into a raw biometric sample that could be recognized by the biometric sensor." PROX_MSFT_003297, Windows Hello biometrics in the enterprise - Windows Security | Microsoft Learn. As such, the biometric data is necessarily stored in tamperproof format unable to subsequently altered.

For at least the foregoing, Windows Hello requires account holders with Windows 10/11 devices to store biometric data of in a tamper proof format that is not capable of being subsequently altered in order to sing in with a biometric gesture. As Microsoft designed Windows Hello and Windows 10/11, Microsoft exercises control and direction over account holders by controlling every step that the software performs and every prompt for user action, to store their biometric data in a tamper proof format written to a storage element on the integrated device that is not capable of being subsequently altered.

the biometric data is selected from a group consisting of a palm print, a retinal scan, an iris scan, a hand geometry, a facial recognition, a signature recognition and a voice recognition

As shown in the below excerpt from Microsoft's depiction of its UPA, use of the Microsoft Authenticator app on integrated devices, such as Android and iOS devices, to authenticate without passwords requires completing a challenge by scanning biometrics (i.e., biometric verification).



6. The user completes the challenge by entering their biometric or PIN to unlock private key.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

"Passwordless sign-in with the Authenticator App" may be accomplished by "[s]ign[ing] in using a mobile phone with fingerprint scan, facial scan or iris recognition". PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn A subset of a palm print is a fingerprint, as a palm print includes at least one

		fingerprint. Accordingly, Microsoft Authenticator Apps causes account holders to select biometric data to complete the challenge from a group consisting of a palm print, a retinal scan, an iris scan, a hand geometry, a facial recognition, a signature recognition and a voice recognition. Also see the following documentation regarding various biometrics in Microsoft's documentation:

Enrolling a Fingerprint

Article • 07/08/2021 • 4 contributors

Feedback

In this article

Enrolling a Fingerprint Overview

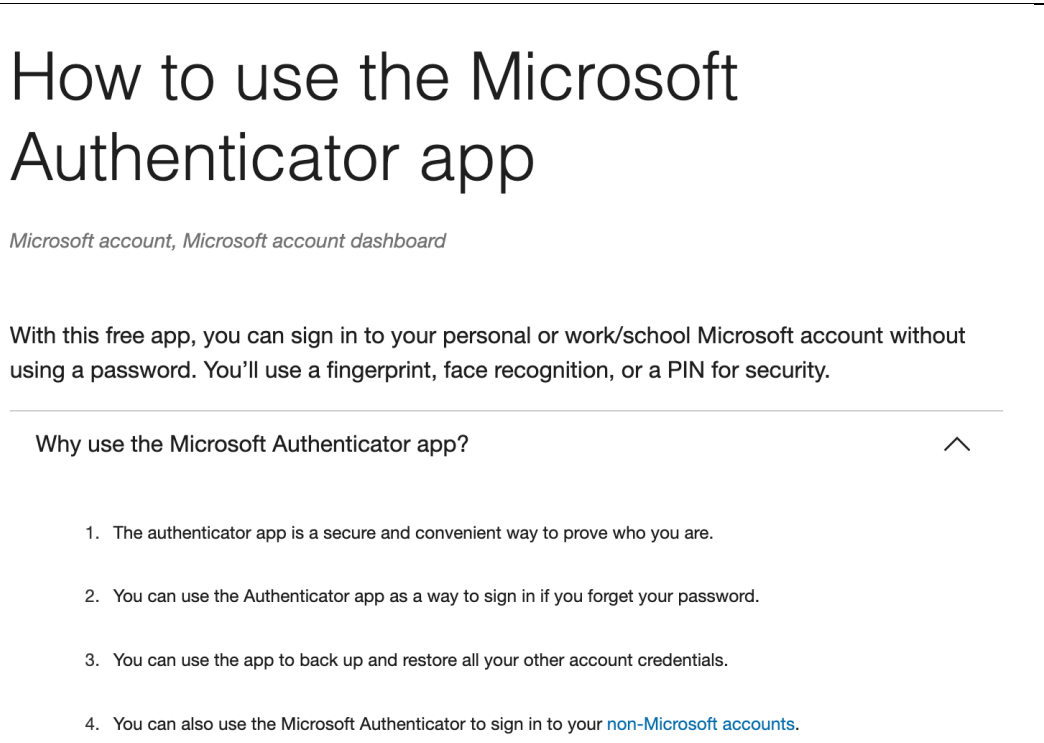
It is only possible for an Android application to leverage fingerprint authentication if the device has already been configured with fingerprint authentication. This guide will discuss how to enroll a fingerprint on an Android device or Emulator. Emulators do not have the actual hardware to perform a fingerprint scan, but it is possible to simulate a fingerprint scan with the help of the Android Debug Bridge (described below). This guide will discuss how to enable screen lock on an Android device and enroll a fingerprint for authentication.

Requirements

To enroll a fingerprint, you must have an Android device or an emulator running API level 23 (Android 6.0).

The use of the Android Debug Bridge (ADB) requires familiarity with the command prompt, and the **adb** executable must be in the PATH of your Bash, PowerShell, or Command Prompt environment.

PROX_MSFT_002995, https://learn.microsoft.com/en-us/xamarin/android/platform/fingerprint-authentication/enrolling-fingerprint



How to use the Microsoft Authenticator app

Microsoft account, Microsoft account dashboard

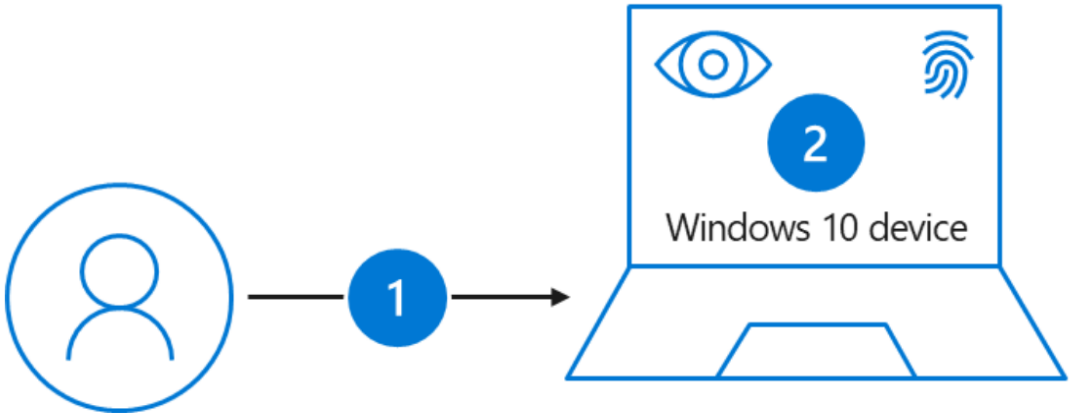
With this free app, you can sign in to your personal or work/school Microsoft account without using a password. You'll use a fingerprint, face recognition, or a PIN for security.

Why use the Microsoft Authenticator app?

1. The authenticator app is a secure and convenient way to prove who you are.
2. You can use the Authenticator app as a way to sign in if you forget your password.
3. You can use the app to back up and restore all your other account credentials.
4. You can also use the Microsoft Authenticator to sign in to your non-Microsoft accounts.

PROX_MSFT_003035, https://support.microsoft.com/en-us/account-billing/how-to-use-the-microsoft-authenticator-app-9783c865-0308-42fb-a519-8cf666fe0acc

Turning to Windows Hello, as shown in the below excerpt, use of the Windows Hello on integrated devices, such as Windows 10/11 laptops and personal computers, to authenticate without passwords requires signing in with a biometric gesture (i.e., biometric verification).



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.

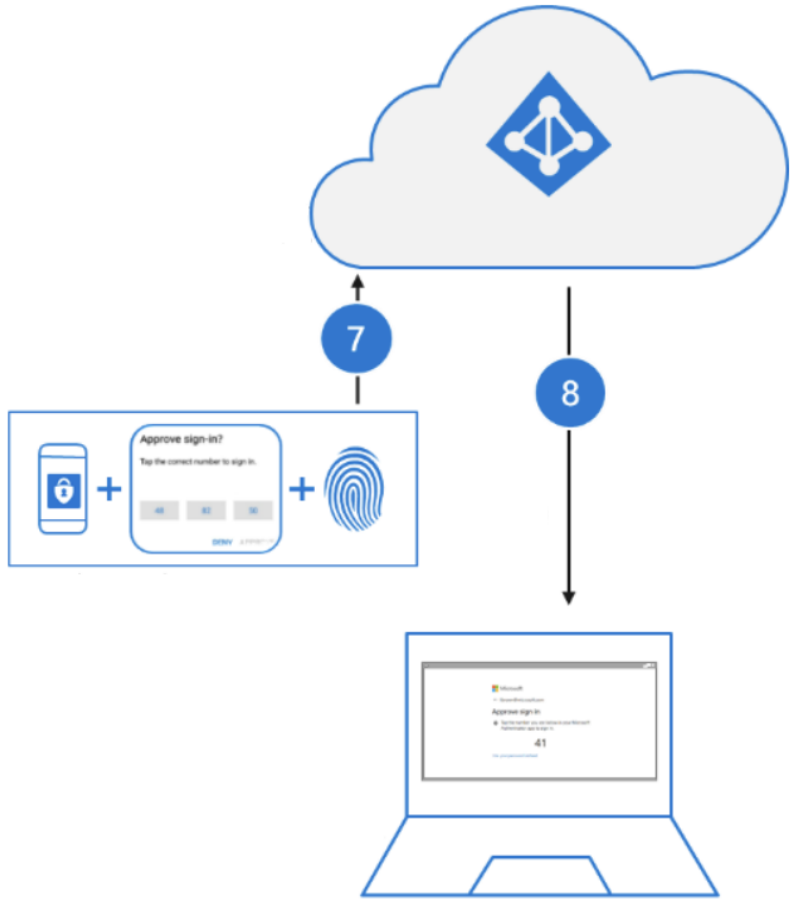
See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

Microsoft states that signing in with Windows Hello may be accomplished by using "biometric recognition (facial, iris, or fingerprint) with Windows devices.". PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn A subset of a palm print is a fingerprint, as a palm print includes at least one fingerprint. Accordingly, Windows Hello causes account holders to select biometric data to use as the sign in gesture from a group consisting of a palm print, a retinal scan, an iris scan, a hand geometry, a facial recognition, a signature recognition and a voice recognition.

persistently storing … a plurality of codes and other data values comprising a device ID code uniquely identifying the integrated device … in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered

Microsoft's UPA enables account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub ("You can now delete your password from your Microsoft account—or set up a new account with no password—and sign in using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys."). The Microsoft Authenticator

		app "turns any iOS or Android phone into a strong, passwordless credential." PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. As shown in the below excerpt from Microsoft's description of its UPA, use of the Microsoft Authenticator App on integrated devices, such as Android and iOS devices, to authenticate without passwords requires that is nonce is signed with a private key followed by public/private key validation. Accomplishing public/private key validation requires selecting the right public to verify the signature generated with the private key. Within Microsoft UPA selecting the right public key is accomplished by sending a stored credential ID stored uniquely identifying account holders integrated iOS or Android device. Accordingly, the Microsoft Authenticator App causes account holders to persistently store a plurality of codes and other data values comprising a device ID code uniquely identifying the integrated device in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered.

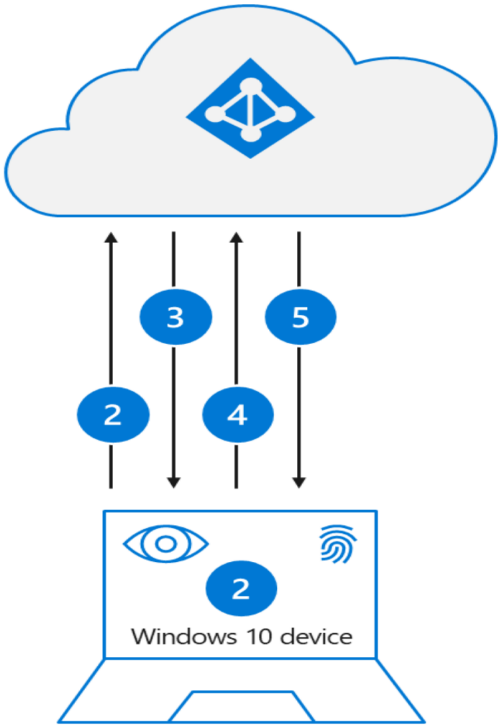


7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

Microsoft explains that the Authenticator app uses key-based authentication technology similar to Windows Hello. ("Microsoft Authenticator uses key-based authentication to enable a user credential that is tied to a device, where the device uses a PIN or biometric. Windows Hello for

		Business uses a similar technology.") Windows Hello is FIDO certified. The FIDO specification incorporates the WebAuthn Specification. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."). Windows Hello, therefore, uses a key-based authentication technology compliant with the WebAuthn and FIDO specifications. Utilizing a similar technology, the MS Authenticator App must use a key-based authentication analogous to the WebAuthn and FIDO protocols. Turning to Windows Hello, as shown in the below excerpt, use of Windows Hello on integrated devices, such as Windows 10/11 laptops and personal computers, to authenticate without passwords requires that is nonce is signed with a private key followed by public/private key validation. Public/private key validation requires selecting the right public key to verify the signature generated with the private key. Within Microsoft UPA, selecting the correct public key is accomplished by sending a stored credential ID uniquely identifying account holders Windows 10/11 device. Accordingly, Windows Hello persistently stores a plurality of codes and other data values comprising a device ID code uniquely identifying the integrated device in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered.



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

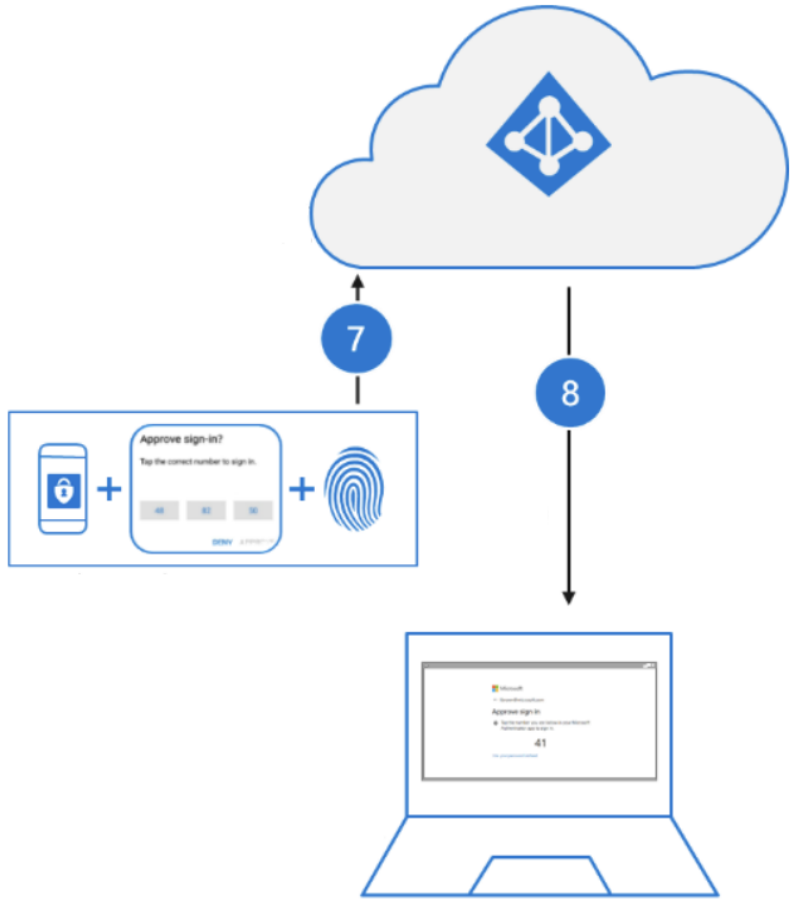
Windows Hello is FIDO certified. The FIDO specification incorporates the WebAuthn Specification. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."). Windows Hello, therefore, uses a key-based authentication technology compliant with the WebAuthn and FIDO specifications.

Under the WebAuthn specification, "compliant authenticators protect public key credentials." PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1. A public key credential refers to a public key credential source, which includes a credential ID. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining "public key credential" and "public key credential source".). The credential ID uniquely identifies its public key credential source. *See* PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a credential ID as "A probabilistically-unique byte sequence identifying a public key credential source and its authentication assertions."). In addition to the credential ID, each public key credential source contains a "credential private key". PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4, (Defining "public key credential source" as data structure including the credential private key and the credential ID.). "The credential private key is bound to a particular authenticator" and part of an asymmetric key pair containing a public key returned to a relying party. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a "credential key pair"); and PROX_MSFT_003293, Passwordless authentication with Azure Active Directory - Microsoft Entra | Microsoft Learn ("Microsoft Authenticator uses key-based authentication to enable a user credential that is tied to a device, where the device uses a PIN or biometric."). Accordingly, a credential ID uniquely identifies a private/public key pair. While some identity providers, such as Google, may permit private keys to be shared across devices, Windows Hello does not. PROX_MSFT_003062, Passkey support on Android and Chrome | Authentication | Google Developers ("Chrome on Windows stores passkeys in Windows Hello, which doesn't sync them to other devices as of January 2023."). Being based on a similar technology as Windows Hello, the Microsoft Authenticator app also does not permit its private keys to be shared. Thus, every public/private key pair will be unique to a specific instance of the Microsoft Authenticator app on a given device. Uniquely identifying device-specific public/private key pairs, the credential ID necessarily uniquely identifies the device, making it a device ID code.

The credential ID is used to retrieve the correct public key during authentication ceremonies. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key."). The public key retrieved is used to verify the signature generated with its matching private key held by the device running the Microsoft

		Authenticator App. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("Using *credentialPublicKey*, verify that *sig* is a valid signature over the binary concatenation of *authData* and *hash*."); and PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn (Detailing authentication with the Microsoft Authenticator app includes signing at nonce with a private key unlocked via biometrics and returning the signed nonce to Microsoft who performs public/private key validation.). By being a part of an asymmetric key pair, the public key corresponds to a private key. Altering a Windows Hello credential ID causes authentication to fail. Being FIDO certified, Windows Hello utilizes the WebAuthn protocol. Credential IDs within the WebAuthn protocol are in a tamper proof format and are unable to be subsequently altered because "all that would happen if an authenticator returns the wrong credential ID, or if an attacker intercepts and manipulates the credential ID, is that the WebAuthn Relying Party would not look up the correct credential public key with which to verify the returned signed authenticator data (a.k.a., assertion), and thus the interaction would end in an error." PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 13.1. As altering the device ID (i.e., Credential ID) causes authentication to fail, the credential ID is necessarily in a tamper proof format unable to be altered. "Passwordless authentication using the Authenticator app follows the same basic pattern as Windows Hello for Business." PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. Following the same pattern, altering the device ID caused to be stored on the device by the Authenticator App should produce the same result of causing authentication to fail. In the alternative, to the extent the above is considered to be a different security implementation or if limited alterations are possible in limited circumstances and given high levels of security permissions, clearance, or hardware-level permission attribution, the "tamper proof format . . . that is unable to be subsequently altered," limitation is met under the doctrine of equivalents on the grounds that the tamper proofing and limitations on alteration required for a Credential ID in the WebAuthn protocol must prevent nearly all alteration by nearly all programs that have not been explicitly approved and, at least, prevent alteration and requires a strict level of physical and/or virtual permission to be altered. Moreover, this limitation could be met under the doctrine of equivalents (if not directly infringed) if alterations break the system's connection with the stored Credential ID and render the Credential ID unusable as this is the same function, way, and result as preventing alterations entirely in the first place. As Microsoft controls the protocol utilized by its Identity Platform, including the ability to sync passkeys across devices, users wanting the benefit of passwordless authentication via the

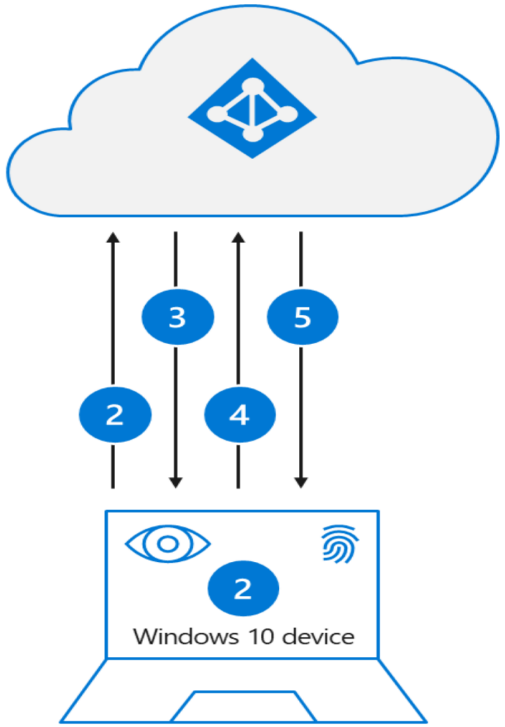
		Microsoft Authenticator app must cause account holders to store on their integrated iOS or Android device a credential ID uniquely identifying the integrated device in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered. persistently storing … a secret decryption value in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered; Microsoft's UPA enables account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub ("You can now delete your password from your Microsoft account—or set up a new account with no password—and sign in using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys."). The Microsoft Authenticator app "turns any iOS or Android phone into a strong, passwordless credential." PROX_MSFT_002830. Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. As shown in the below excerpt, use of the Microsoft Authenticator app on integrated devices, such as Android and iOS devices, to authenticate without passwords requires that is nonce is signed with a private key followed by public/private key validation.



7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

As shown in the below, use of Windows Hello on integrated devices, such as Windows 10/11 laptops and personal computers, to authenticate without passwords requires that is nonce is signed with a private key followed by public/private key validation.

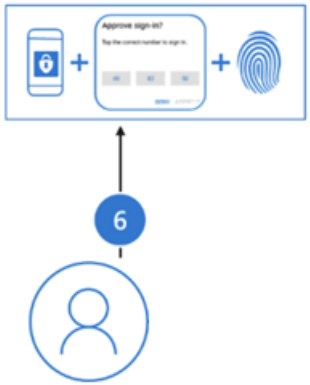


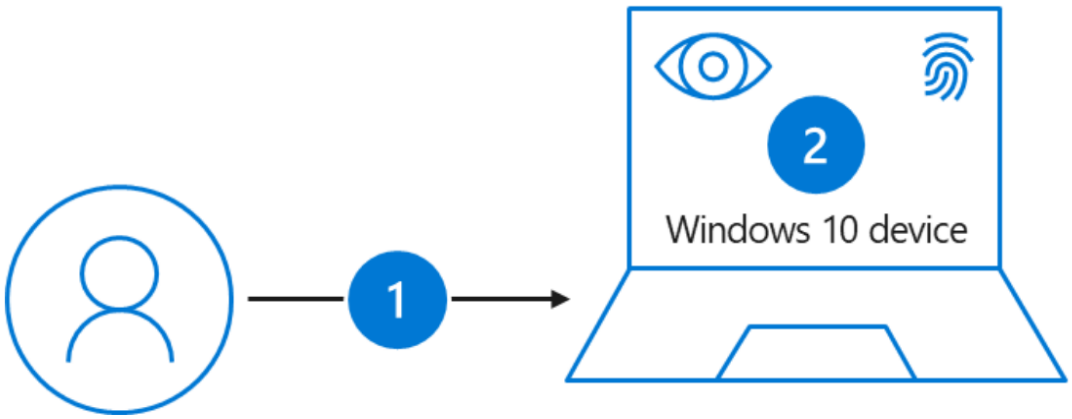
1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.

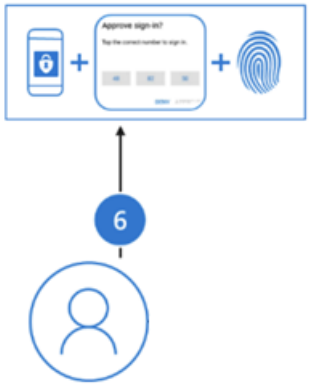
Id.

Private keys are secret decryption values.

If the Authenticator App has the private key, the public key must be held by Microsoft's Identity Platform; something Microsoft confirms. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn ("Azure AD performs public/private key validation and returns a token"). By being part of a pair split between the authenticator and Microsoft's Identity Platform, the private key secret decryption value is in a tamper proof format unable to be subsequently altered. Utilizing a protocol similar to WebAuth standard employed

		by Windows Hello, during authentication, the appropriate public key is located utilizing the device ID supplied by the authenticator to verify the signature generated using the Authenticator App's private key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash.") If the Authenticator App's private key was altered, it would generate a signature that could not be verified with the public key held by Microsoft Identity Platform. Such would cause the authentication ceremony to fail. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("22. If all the above steps are successful, continue with the authentication ceremony as appropriate. Otherwise, fail the authentication ceremony."). Accordingly, by being part of a pair divided between the Microsoft Identity Platform and the Authenticator App (or Windows Hello), the private key is in a tamper proof format unable to be subsequently altered. As Microsoft controls the protocol utilized by its Identity Platform, including the user of public/private key validation, users wanting the benefit of passwordless authentication via the Microsoft Authenticator App (and/or Windows Hello) must cause account holder to store on their device a secret decryption value in a tamper proof format written to a storage element on the integrated device that is unable to be subsequently altered.
	responsive to receiving a request for a biometric verification of the user, receiving scan data from a biometric scan;	As shown in the below excerpt from Microsoft's description of its UPA, use of the Microsoft Authenticator App on integrated devices, such as Android and iOS devices, to authenticate without passwords requires completing a challenge by scanning biometrics (i.e., biometric verification).  6. The user completes the challenge by entering their biometric or PIN to unlock private key.

		See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless. Entering their biometric necessarily requires receiving scan data from a biometric scan using a device, such as an iOS or Android device. Accordingly, Microsoft utilizes the biometric sensors in smartphones, tablets, and computers to receive data from those biometric scans. Turning to Windows Hello, signing in with a biometric gesture necessarily requires receiving scan data from a biometric scan. Accordingly, Microsoft utilizes the biometric sensors on laptops and personal computers to receive data from those biometric scans.  1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*. *See* PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.
	comparing the scan data to the biometric data to determine whether the scan data matches the biometric data;	As shown in the below excerpt from Microsoft's own depiction of its UPA, use of the Microsoft Authenticator App on integrated devices, such as Android and iOS devices, to authenticate without passwords requires completing a challenge by scanning biometrics (i.e., biometric verification).

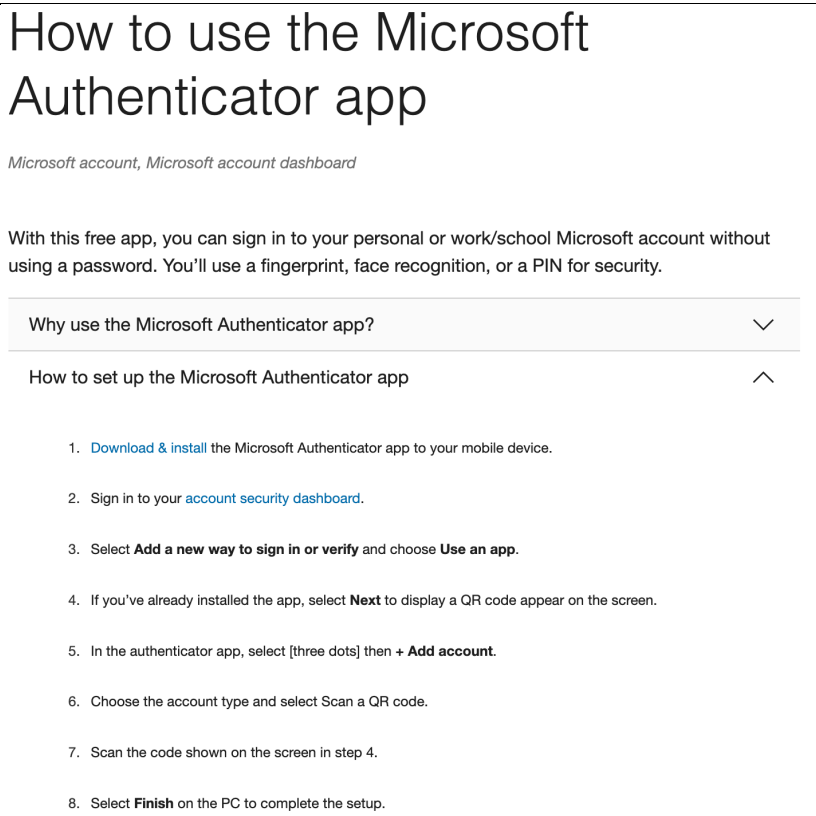


6. The user completes the challenge by entering their biometric or PIN to unlock private key.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

Determining if the user successfully completed a "challenge" requires the entered biometrics be compared against stored biometric data to determine whether the entered data matches the stored biometric data. Without such a comparison, it would not be a "challenge." Accordingly, the challenge necessarily requires comparing the scan data from a biometric scan with the data that had been stored previously as the template or baseline of the authenticated user.

Additional Microsoft documentation regarding the Authenticator app makes clear that a fingerprint or face recognition may be used for security:



How to use the Microsoft Authenticator app

Microsoft account, Microsoft account dashboard

With this free app, you can sign in to your personal or work/school Microsoft account without using a password. You'll use a fingerprint, face recognition, or a PIN for security.

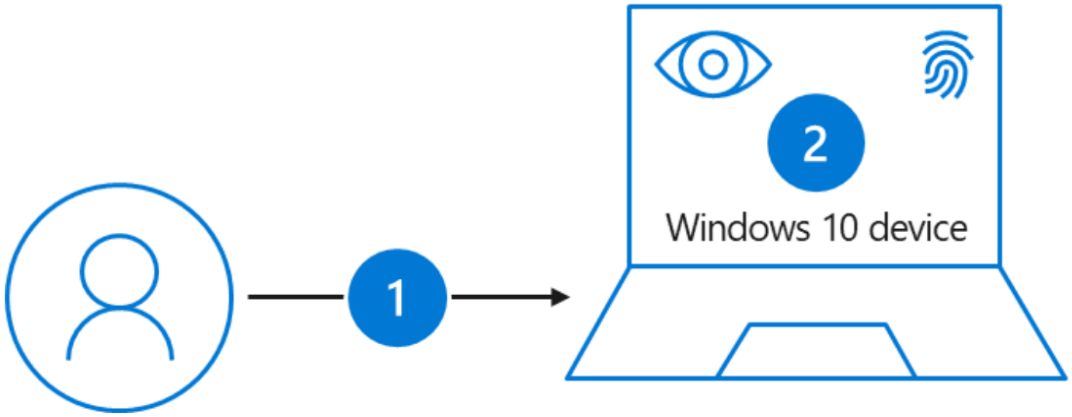
Why use the Microsoft Authenticator app?

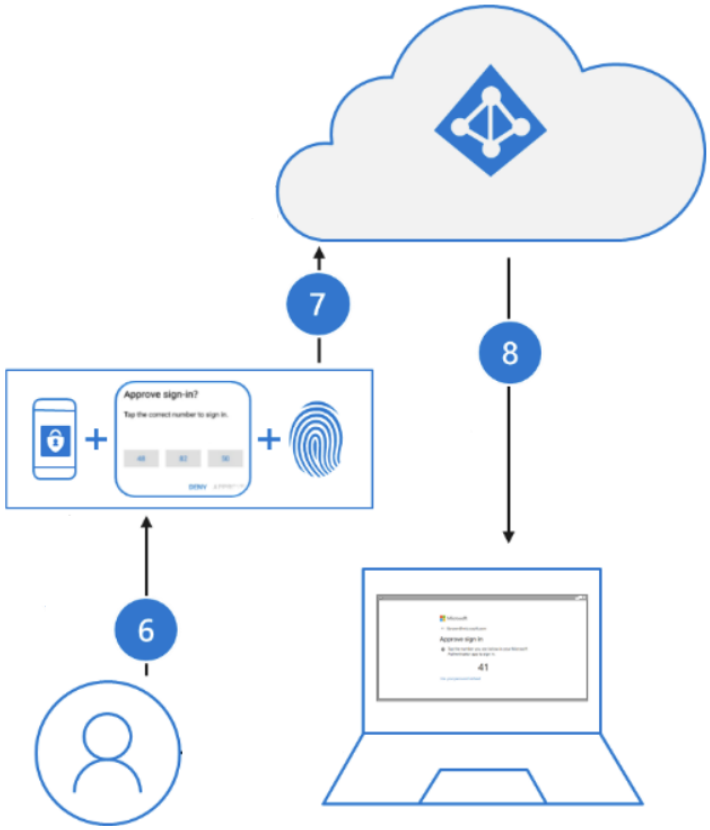
How to set up the Microsoft Authenticator app

1. Download & install the Microsoft Authenticator app to your mobile device.
2. Sign in to your account security dashboard.
3. Select **Add a new way to sign in or verify** and choose **Use an app**.
4. If you've already installed the app, select **Next** to display a QR code appear on the screen.
5. In the authenticator app, select [three dots] then **+ Add account**.
6. Choose the account type and select Scan a QR code.
7. Scan the code shown on the screen in step 4.
8. Select **Finish** on the PC to complete the setup.

PROX_MSFT_003035, https://support.microsoft.com/en-us/account-billing/how-to-use-the-microsoft-authenticator-app-9783c865-0308-42fb-a519-8cf666fe0acc

Turning to Windows Hello, signing in with a biometric gesture requires the entered biometrics be compared against stored biometric data to determine whether the entered data matches the stored biometric data.

		 1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
	responsive to a determination that the scan data matches the biometric data, wirelessly sending one or more codes from the plurality of codes and the other data values for authentication by an agent that is a third-party trusted authority possessing a list of device ID codes uniquely identifying legitimate integrated devices, wherein the one or more codes and other data values includes the device ID code	responsive to a determination that the scan data matches the biometric data, wirelessly sending

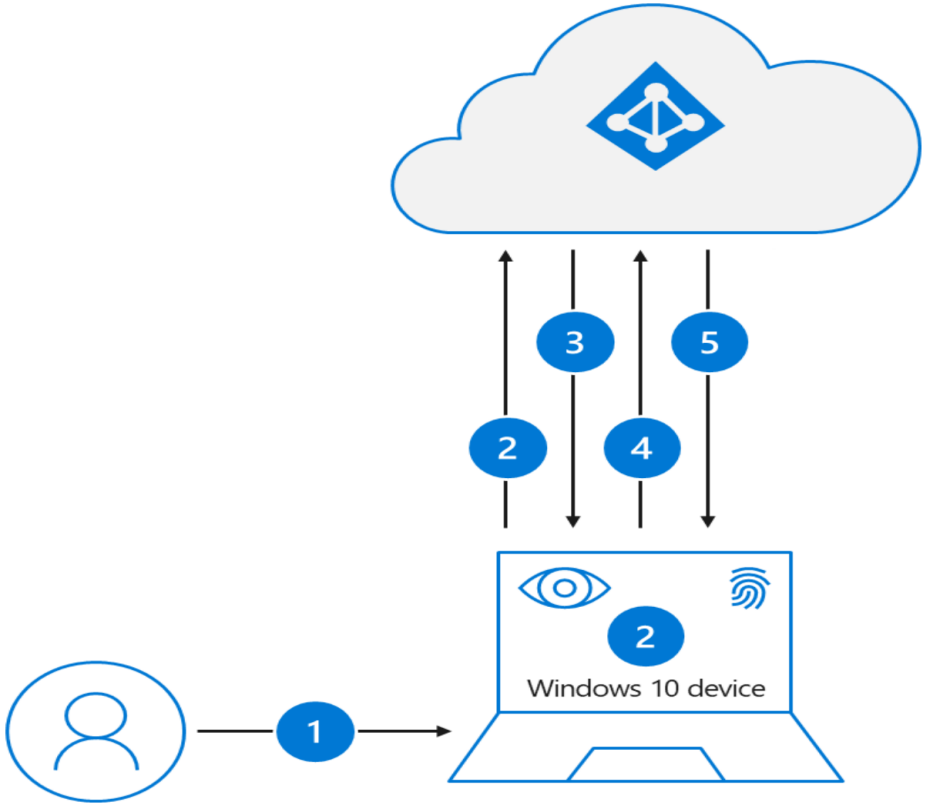


6. The user completes the challenge by entering their biometric or PIN to unlock private key.
7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

As shown in the above excerpt, after the user has completed the biometric challenge, a signature is generated with the Authenticator App's private key and sent to Microsoft Identity Platform for verification. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn ("The nonce is signed with the private key and sent back to Azure

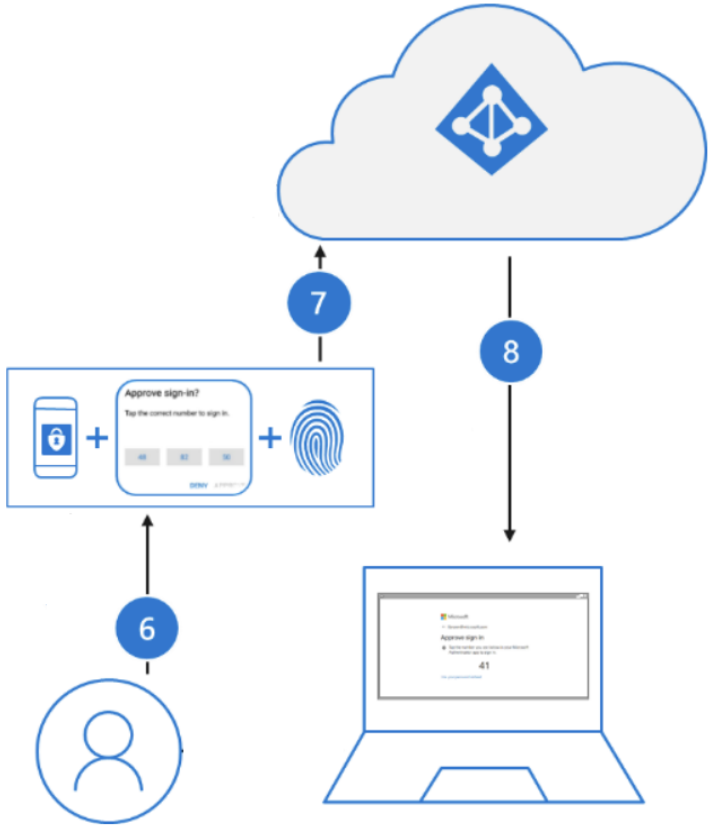
		AD.”). The Microsoft Authenticator app operates on either iOS or Android devices. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. (The Microsoft Authenticator app "turns any iOS or Android phone into a strong, passwordless credential.”) As iOS and Android devices may include cellular, Wi-Fi, and/or Bluetooth capability, anything sent to Microsoft Identity Platform will be sent wirelessly – either via the cellular network, Wi-Fi, or Bluetooth. Similarly, in Windows Hello, after the user has signed in with a biometric gesture, a signature is generated with Windows Hello’s private key and sent to Microsoft Identity Platform for verification. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn (“The Cloud AP signs the nonce using the user’s private key and returns the singed nonce to the Azure AD.”). Windows Hello operates on Windows 10/11 laptops and personal computers, which connect to the internet via Wi-Fi. As Wi-Fi is a wireless protocol, anything sent to Microsoft Identity Platform via Wi-Fi will be sent wirelessly.



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

responsive to a determination that the scan data match the biometric data, … sending one or more codes and other values from the plurality of codes and other data value for authentication … wherein the one or more codes includes the device ID code

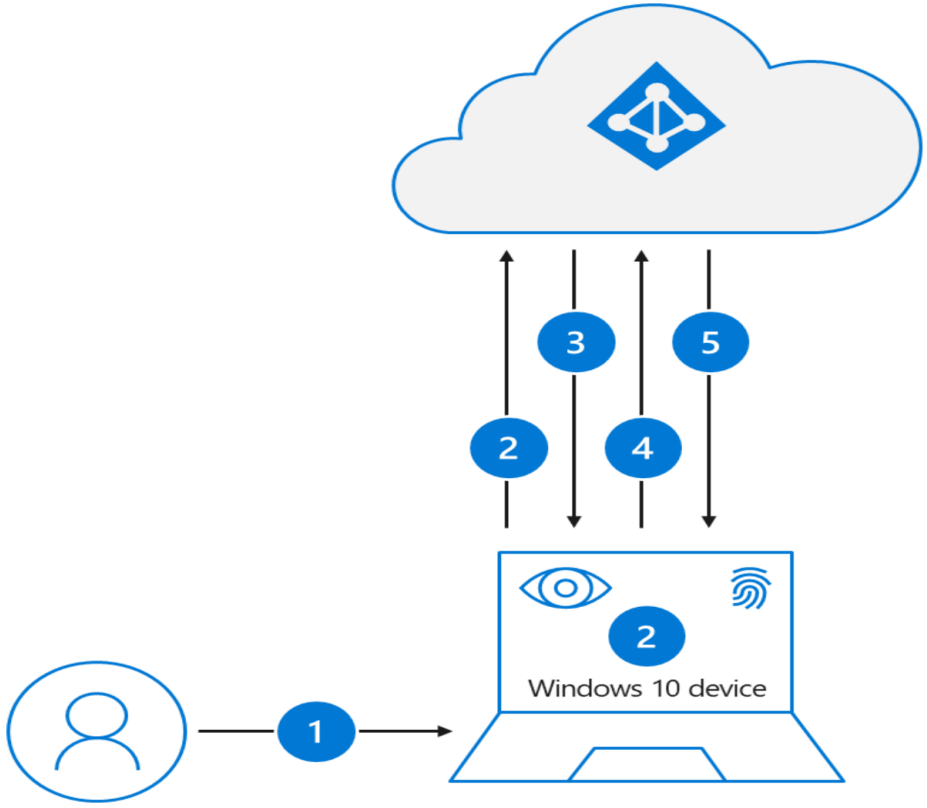


6. The user completes the challenge by entering their biometric or PIN to unlock private key.
7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

As shown in the above excerpt from Microsoft's description of its UPA, after the user has completed the biometric challenge, a signature is generated with the Authenticator App's private key and sent to Microsoft Identity Platform for verification. PROX_MSFT_002830, Azure

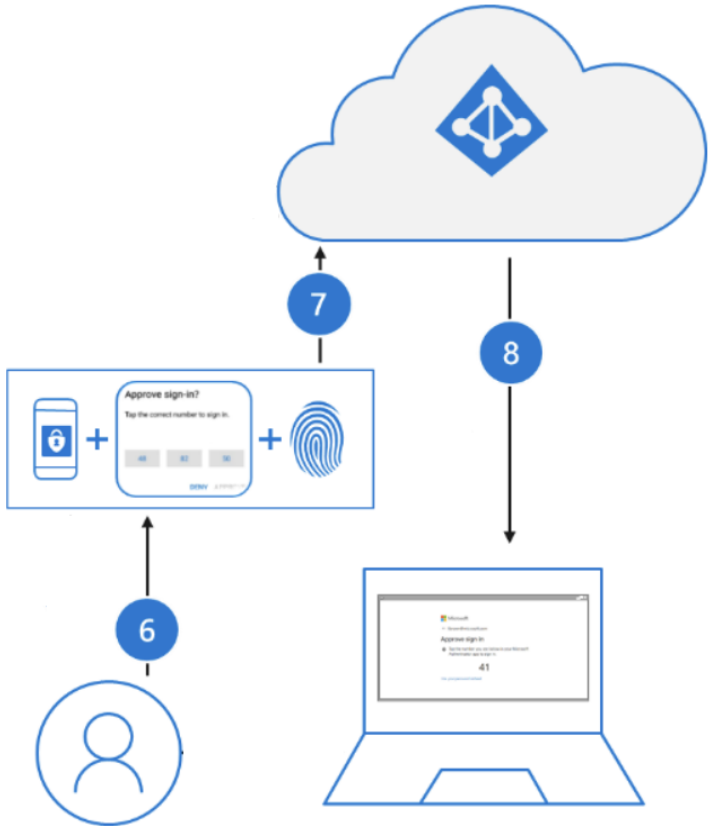
		Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn ("The nonce is signed with the private key and sent back to Azure AD. Azure AD performs/public private key validation."). Turning to Windows Hello, as shown in the below excerpt, after the user has signed in with a biometric gesture, a signature is generated with Windows Hello's private key and sent to Microsoft Identity Platform for verification. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn ("The Cloud AP signs the nonce using the user's private key and returns the singed nonce to the Azure AD.").



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.

As previously noted, passwordless authentication using the Authenticator App follows the same basic pattern as FIDO-certified Windows Hello. FIDO incorporates the WebAuthn Protocol. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1.1 ("Along with the Web Authentication API itself, this specification defines a request-response cryptographic protocol—the WebAuthn/FIDO2 protocol—between a WebAuthn Relying Party server and an authenticator, where the Relying Party's request consists of a challenge and other input data supplied by the

		Relying Party and sent to the authenticator.") Accordingly, the challenge sent by the Authenticator App is the equivalent of a FIDO server challenge sent to the authenticator with a WebAuthn authenticatorGetAssertion request. The response to the request, therefore, will contain a signature signed by the Authenticator App's private key and a credential ID (i.e., device ID) identifying which public key to use to verify the signature. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 (Detailing the response to authenticationGetAssertion as containing "PublicKeyCredentialDescriptor structure containing the credential identifier whose private key was used to generate the assertion" and "an assertion signature".); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 6 (Defining an "assertion signature" as "an assertion signature asserts that the authenticator possessing a particular credential private key has established, to the best of its ability, that the user requesting this transaction is the same user who consented to creating that particular public key credential."). Upon receiving the response, the WebAuthn/FIDO server will use the device ID (i.e., credential ID) to locate the appropriate public key to verify the signature generated with the private key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash.") Accordingly, a FIDO/WebAuthn server operated by Microsoft Identity will authenticate a device ID (i.e., credential ID) sent in response to successful completion of a biometric challenge. As Microsoft controls the protocol utilized by its Identity Platform, including the use of public/private key pair identified with a credential ID for validation, users wanting to sign in to their Microsoft Account without a password utilizing Microsoft Authenticator App (or Windows Hello) must send a device ID code for authentication upon completion of the biometric challenge. responsive to a determination that the scan data matches the biometric data, wirelessly sending … for authentication by an agent that is a third-party trusted authority

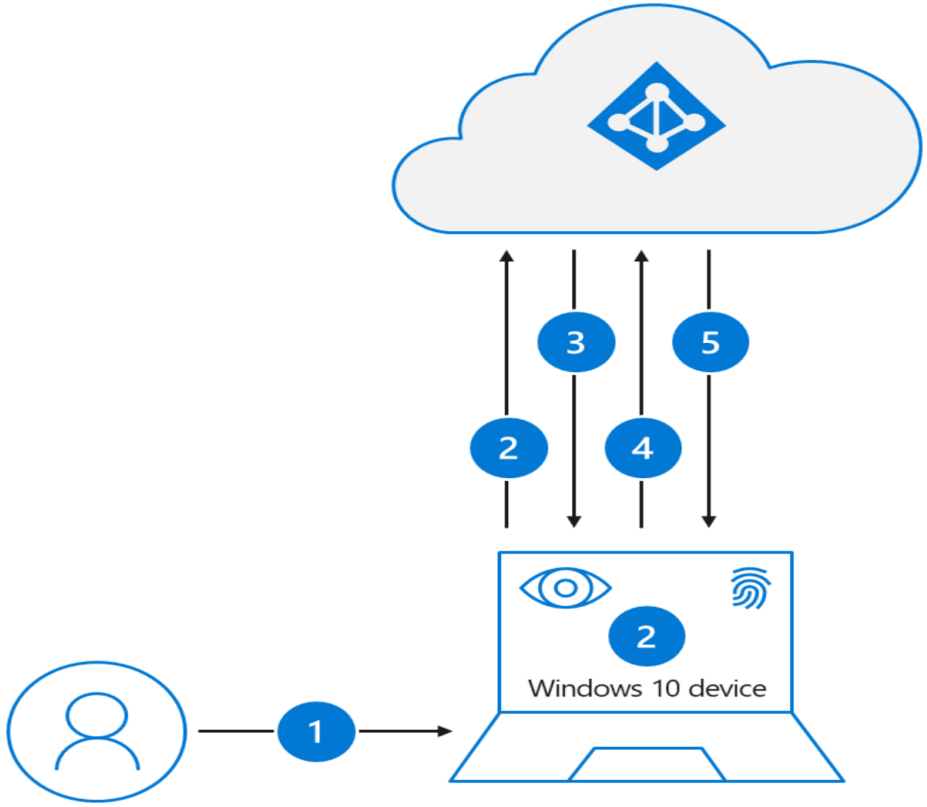


6. The user completes the challenge by entering their biometric or PIN to unlock private key.
7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

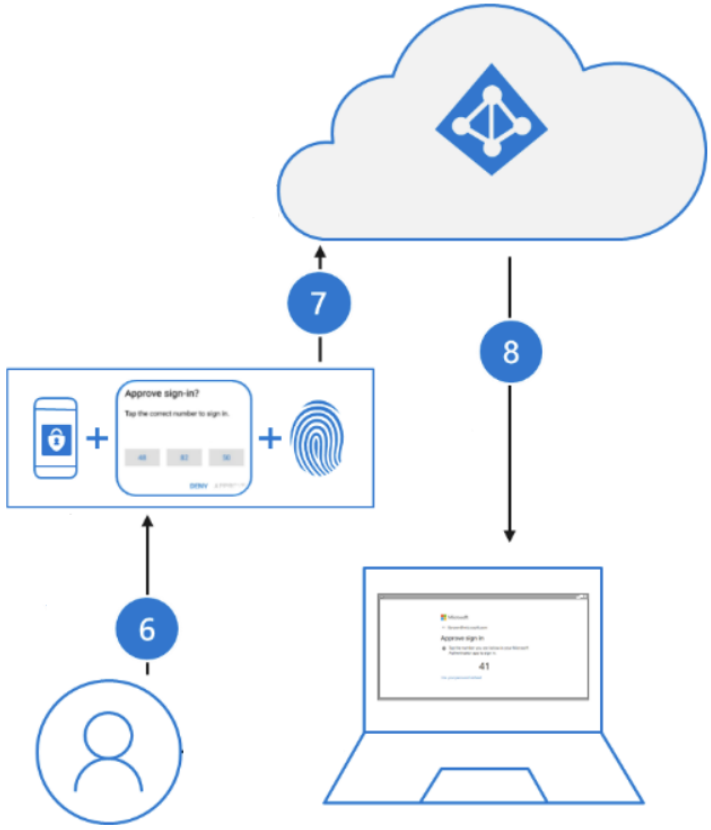
As shown in the above excerpt and detailed *supra*, upon successful completion of the biometric challenge, a device ID and signature is sent from the Authenticator App to Microsoft Identity Platform for authentication.

Turning to Windows Hello, as shown in the below excerpt and detailed *supra*, upon signing in with a biometric gesture, a device ID and signature is sent from Windows Hello to Microsoft Identity Platform for authentication.

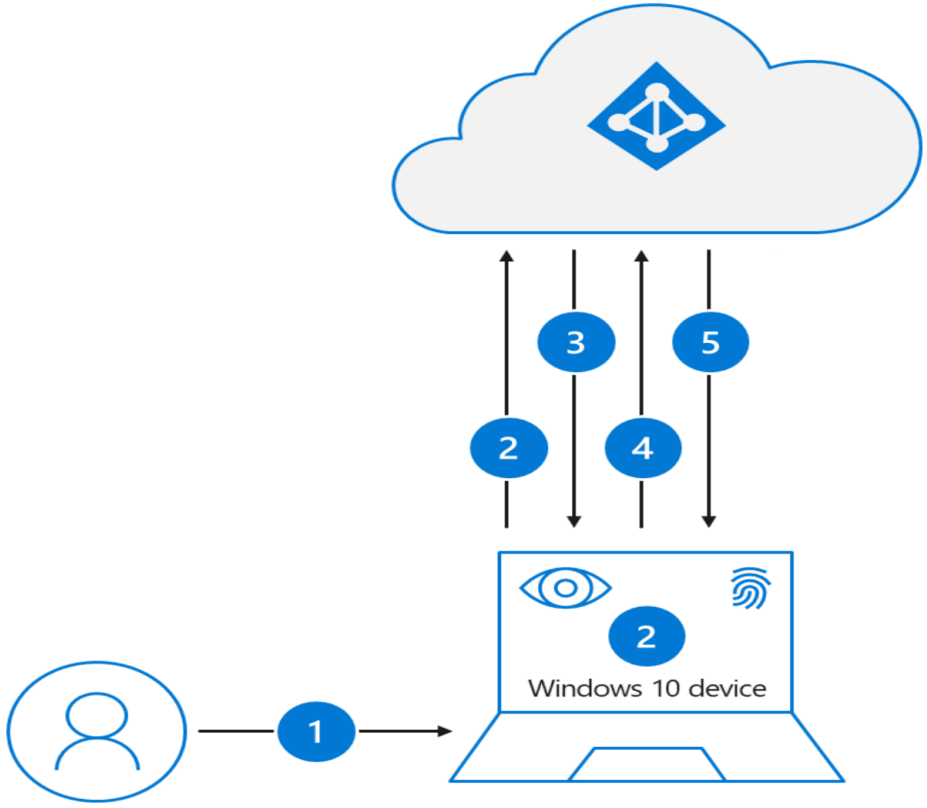


1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.

		The Microsoft Identity Platform utilizes OpenID Connect and Microsoft encourages developers to gain an understanding of the protocol and concepts to add authentication to applications. PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("Knowing about OAuth or OpenID Connect (OIDC) at the protocol level isn't required to use the Microsoft identity platform. However, you'll encounter protocol terms and concepts as you use the identity platform to add authentication to your apps."). Within the protocol, Microsoft identifies its identity platform as an "authorization server" managing "trust relationships" and issuing security tokens applications and APIs use for granting access (i.e. authorization) and authentication. PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("The identity platform is the authorization server. Also called an identity provider or IdP, it securely handles the end-users information, their access, and the trust relationships between the parties in the auth flow. The authorization server issues the security tokens your apps and APIs use for granting, denying, or revoking access to resources (authorization) after the user has signed in (authenticated)."). One of the tokens issued by Microsoft's Identity Platform is an ID Token used when in signing in users. PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("ID tokens are issued by the authorization server to the client application. Clients use ID tokens when signing in users and to get basic information about them."). Accordingly, Microsoft describes its Identity Platform as a third party that operates as a trusted authority for authentication. possessing a list of device ID codes uniquely identifying legitimate integrated devices As noted supra, the FIDO server within the Microsoft Identity Platform uses a credential ID operating as a device ID to verify a signature generated with a private key bound to the iOS or Android device running the MS Authenticator App. Such action is made possible by registering the credentials with an account and associating the account with the credential ID and credential public key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.1. (Detailing the registration ceremony as including "register the new credential with the account that was denoted in options.user" and "Associate the user's account with the credentialId and credentialPublicKey in authData.attestedCredentialData".). Accordingly, the Microsoft Identity Platform maintains a listing of legitimate device IDs.

	responsive to authentication of the one or more codes and the other data values by the agent, receiving an access message from the agent allowing the user access to an application, wherein the application is selected from a group consisting of a casino machine, a keyless lock, a garage door opener, an ATM machine, a hard drive, computer software, a web site and a file.	responsive to authentication of the one or more codes and the other data values by the agent, receiving an access message from the agent allowing the user access to an application  6. The user completes the challenge by entering their biometric or PIN to unlock private key. 7. The nonce is signed with the private key and sent back to Azure AD. 8. Azure AD performs public/private key validation and returns a token. *See* PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless.

		As shown in the above excerpt, after the public/private key validation of the credential ID (as the device ID code of the one more codes and other data values) by Azure Active Directory (i.e. acting as an agent that is a third party trusted authority) pictured as a cloud, a token is returned. The specific token returned will be an OpenID Connect ID token and/or access token, both of which operate as access messages allowing the account holder to access an application. Turning to Windows Hello, as shown in the below excerpt, after the public/private key validation of the credential ID serving as the device ID code of the one more codes and other data values by Microsoft Identity acting as an agent that is a third party trusted authority, a primary refresh token is returned which operates as access messages allowing the account holder to access an application



1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*.
2. The Cloud AP provider requests a nonce (a random arbitrary number that can be used just once) from Azure AD.
3. Azure AD returns a nonce that's valid for 5 minutes.
4. The Cloud AP provider signs the nonce using the user's private key and returns the signed nonce to the Azure AD.
5. Azure AD validates the signed nonce using the user's securely registered public key against the nonce signature. Azure AD validates the signature and then validates the returned signed nonce. When the nonce is validated, Azure AD creates a primary refresh token (PRT) with session key that is encrypted to the device's transport key and returns it to the Cloud AP provider.

While OpenID Connect is a federation protocol, it is compatible and complementary with FIDO. "The value of a FIDO authentication capability is amplified by a federated system, where the federation system extends the benefits of a FIDO authentication to applications and services without requiring FIDO to be directly integrated with those applications." PROX_MSFT_003006, Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf (fidoalliance.org), page 4. When a federated system, such as OpenID Connect, is combined with FIDO, the OpenID Provider (OP) sends a FIDO server challenge to an authenticator, which is

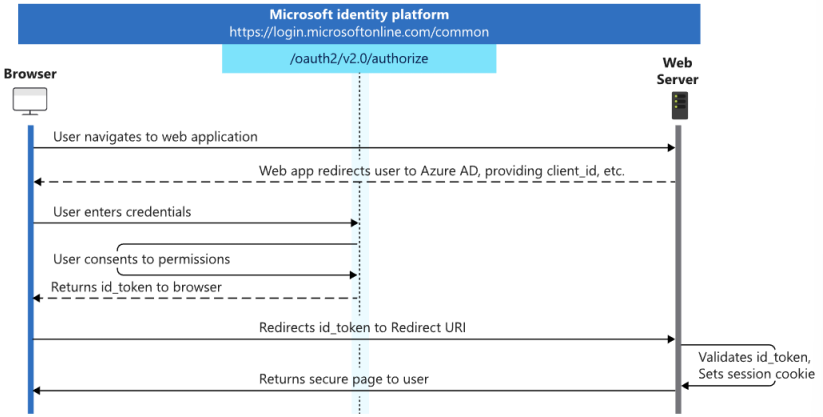
returned as the FIDO Authentication response. PROX_MSFT_003006, Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf (fidoalliance.org), page 10 (Detailing the combined use FIDO and federation entails "3… The Authentication Authority determines that the FIDO authentication specified in the previous request is both relevant… 4. The Authentication Authority sends a FIDO Server challenge… 6. FIDO Client returns FIDO authentication response… 7. FIDO Server validates the FIDO response and reports same to Authentication Authority which looks up the user and then redirects the User Agent back to the Application Provider with an authentication assertion.").

FIDO incorporates the WebAuthn Protocol. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1.1 ("Along with the Web Authentication API itself, this specification defines a request-response cryptographic protocol—the WebAuthn/FIDO2 protocol—between a WebAuthn Relying Party server and an authenticator, where the Relying Party's request consists of a challenge and other input data supplied by the Relying Party and sent to the authenticator.") Accordingly, the FIDO server challenge sent will a be WebAuthn authenticatorGetAssertion request. The response to the request received will contain a signature generated by the authenticator's private key and a credential ID. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 (Detailing the response to authenticationGetAssertion as containing "PublicKeyCredentialDescriptor structure containing the credential identifier whose private key was used to generate the assertion" and "an assertion signature".); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 6 (Defining an "assertion signature" as "an assertion signature asserts that the authenticator possessing a particular credential private key has established, to the best of its ability, that the user requesting this transaction is the same user who consented to creating that particular public key credential."). Upon receiving the response, the WebAuthn/FIDO server will use the credential ID to locate the appropriate public key to verify the signature generated with the private key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash.") Accordingly, a FIDO/WebAuthn server operated by Microsoft Identity Platform will authenticate a device ID (i.e., credential ID). After validating the device ID, the FIDO/WebAuthn portion of Identity Platform "redirects the user agent back to the Application Provider with an authentication assertion". PROX_MSFT_003006,

Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf (fidoalliance.org), page 10.

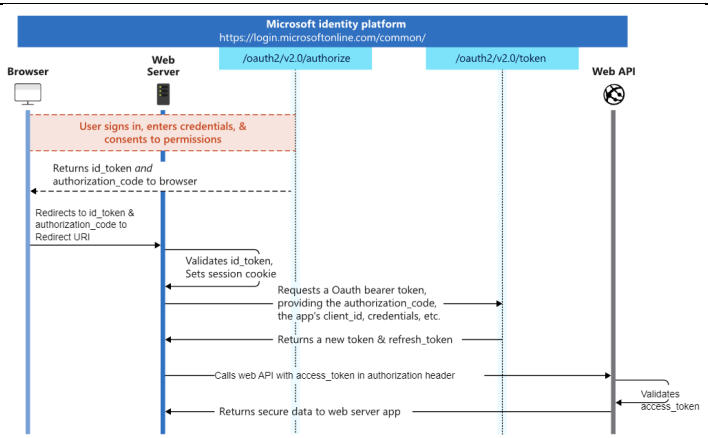
As the Microsoft Identity Platform utilizes OpenID Connect, the authentication assertion will be an ID Token. OpenID Connect defines an ID Token as "a security token that contains Claims about the Authentication of an End-User by an Authorization Server when using a Client, and potentially other requested Claims."

The ID Token received from Microsoft's Identity platform enables access to an application. Such is clearly indicated in the below flow chart of Microsoft's implementation of OpenID Connect. As can be seen, at the end of the process, the ID Token received is exchanged for access to an application.



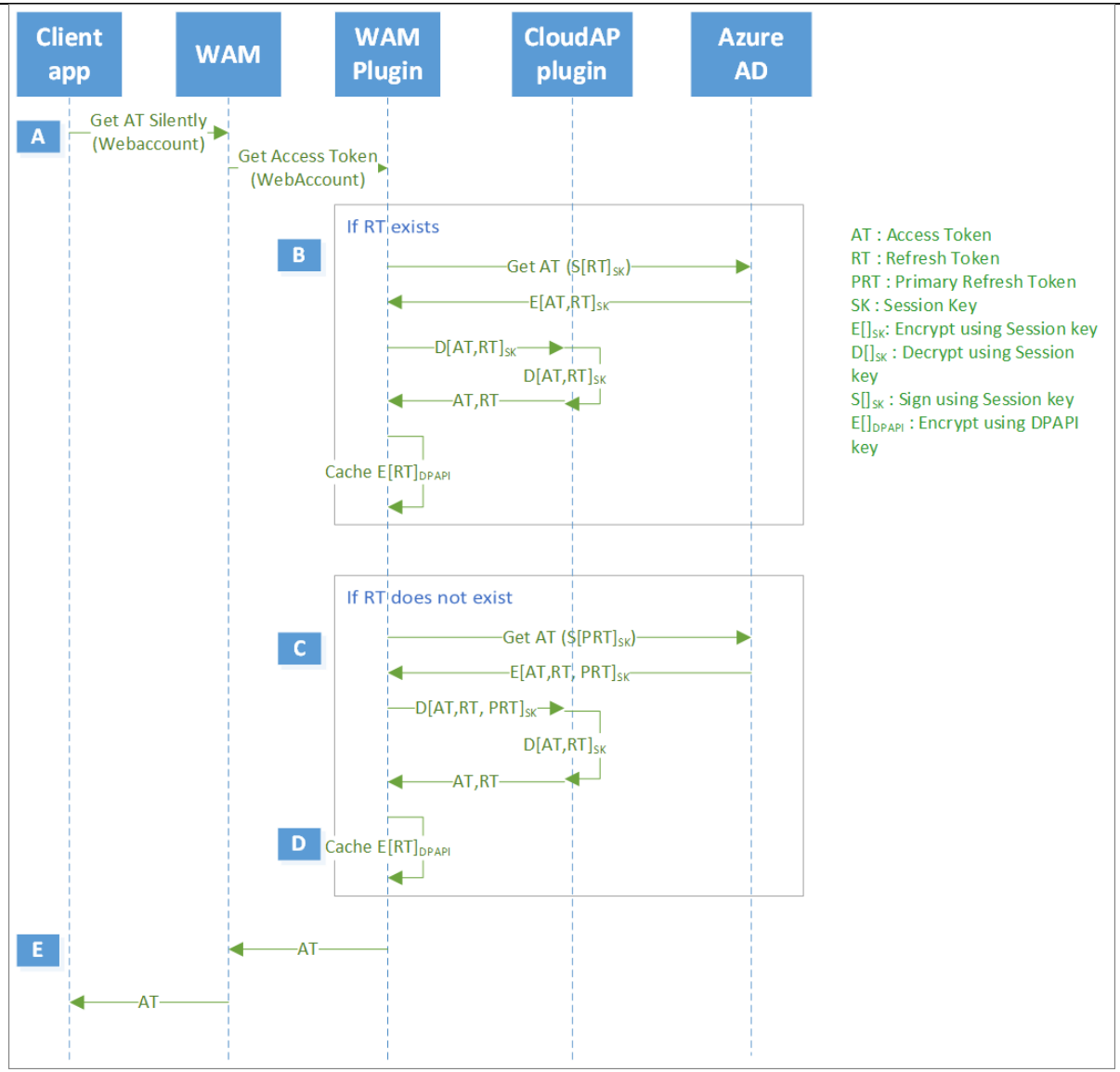
PROX_MSFT_003047, OpenID Connect (OIDC) on the Microsoft identity platform - Microsoft Entra | Microsoft Learn

The ID Token is thus an access message provided by Microsoft Identity operating as third party trusted authority and allowing access to an application. Should the user also require an access token to access further features of the application, such as files containing stored data, the ID token can be exchanged for an access token utilizing the flow detailed below.



PROX_MSFT_003047, [OpenID Connect (OIDC) on the Microsoft identity platform - Microsoft Entra | Microsoft Learn]

When utilizing Windows Hello, the authentication assertion will be the Primary Refresh Token which can be exchanged for Refresh Tokens, which in turn is exchanged for an Access Tokens. Such a process is shown in the below figure.



PROX_MSFT_003302, Primary Refresh Token (PRT) and Azure Active Directory - Microsoft Entra | Microsoft Learn

When the user attempts to access an application (A), the application initiates a token request to WAM. It is then (B) determined if refresh token for the application is available, if so, the refresh token is used to request an access token. The access token is then used to provide access to the

		application. PROX_MSFT_003294, Tokens and claims overview - Microsoft Entra | Microsoft Learn ("Resources validate access tokens to grant access to a client application."). If, however, a refresh token is not available, (C) WAM uses the uses the primary refresh token to request both an access token and refresh token. As the Primary Refresh Token received from Microsoft's Identity Platform during initial login can be exchanged for an Access Token to be used gain access to an application, the primary refresh token is thus an access message from the agent allowing the user access to an application. wherein the application is selected from a group consisting of a casino machine, a keyless lock, a garage door opener, an ATM machine, a hard drive, computer software, a web site and a file. As noted above, Microsoft utilizes Open ID Connect, which includes a client that Microsoft identifies as including web apps and web APIs, both of which represent software, or single page within the browser (i.e., website). PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("The client could be a web app running on a server, a single-page web app running in a user's web browser, or a web API that calls another web API. You'll often see the client referred to as client application, application, or app."). Accordingly, once the account holder has complete passwordless authentication via the Microsoft Authenticator App, they can access Office 365 applications and files, Xbox Live, synced credit cards, and other applications hosted by the Microsoft's resource servers (i.e., clients) and the resource servers of Microsoft Identity's customers.
2	The method of claim 1, wherein the one or more codes and other data values are transmitted to the agent over a network	Data sent to Microsoft Identity Platform will be sent over internet which is network.
3	The method of claim 1, further comprising: registering an age verification for the user in association with the device ID code.	Microsoft Family, which interfaces with the UPA, enables parents to "designate the age limit for content [a family member] will have permission to access", such as apps and games. PROX_MSFT_003022, Filter apps and games with Microsoft Edge - Microsoft Support; see also PROX_MSFT_003084, Set an age limit for content on Xbox | Xbox Support (Detailing how to set content restriction by age for members of an Xbox family group.).

Filter apps and games with Microsoft Edge

Microsoft account

Enjoy peace of mind while younger family members explore online by establishing content filters. Content filters are an effective way to prevent access to mature web content, apps, and games. Set individual age limits within a family group to ensure family members only view age-appropriate material. App and game filters only work on Windows and Xbox devices operating Microsoft Edge - try it now!

PROX_MSFT_003022, https://support.microsoft.com/en-us/account-billing/filter-apps-and-games-with-microsoft-edge-da38c83a-0717-31ea-b6f2-6cdd95639189

Set content restrictions by age

Organizers of an Xbox family group can set content restrictions by using the Xbox console or the Xbox Family Settings app. To learn more about using this feature with the Xbox Family Settings app, see:

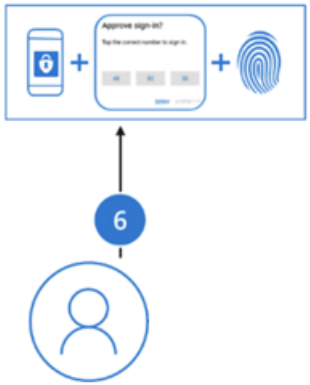
Manage a member's content in the Xbox Family Settings app

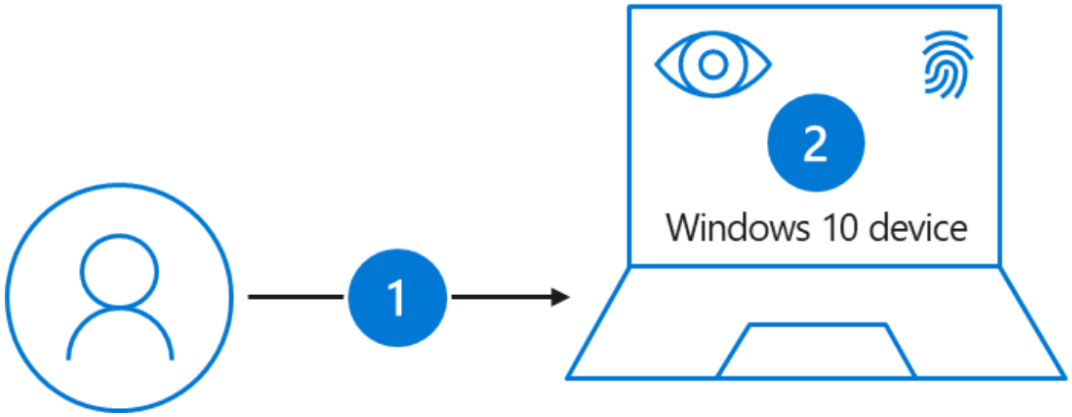
To configure content restrictions on the Xbox console:

1. Press the **Xbox** button to open the guide, and then go to **Profile & system** > **Settings** > **Account** > **Family settings**.
2. Select **Manage family members**, and then choose the member whose settings you want to update.
3. Under **Access to content**, select the desired age-level.

> **Note** By default, a member's age limits are set to the birthdate that's associated with their Microsoft account.

PROX_MSFT_003084, https://support.xbox.com/en-US/help/family-online-safety/online-safety/set-age-limit#:~:text=To%20configure%20content%20restrictions%20on%20the%20Xbox%20console%3A,Under%20Access%20to%20content%2C%20select%20the%20desired%20age-level.

5	The method of claim 1, wherein the biometric data and the scan data are both based on a fingerprint scan by the user.	As shown in the below excerpt from Microsoft's description of its UPA, use of the Microsoft Authenticator app on integrated devices, such as Android and iOS devices, to authenticate without passwords requires completing a challenge by scanning biometrics (i.e., biometric verification).  6. The user completes the challenge by entering their biometric or PIN to unlock private key. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. Microsoft states that "Passwordless sign-in with the Authenticator App" may be accomplished by "[s]ign[ing] in using a mobile phone with fingerprint scan, facial scan or iris recognition". PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. Signing in with a fingerprint necessities that both the scan data and the biometric data are based on a fingerprint scan of the account holder.

		 1. A user signs into Windows using biometric or PIN gesture. The gesture unlocks the Windows Hello for Business private key and is sent to the Cloud Authentication security support provider, referred to as the *Cloud AP provider*. PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn. Microsoft itself states that signing in with Windows Hello may be accomplished by using "biometric recognition (facial, iris, or fingerprint) with Windows devices.". PROX_MSFT_002830, Azure Active Directory passwordless sign-in – Microsoft Entra | Microsoft Learn. Signing in with a fingerprint necessities that both the scan data and the biometric data are based on a fingerprint scan of the account holder.

'730 Patent Claim[6]	**Accused Instrumentality And Where Each Claim Element Is Found[7]** **FIDO Implementation**

[6] All PICS set forth herein for any independent patent claims are hereby incorporated by reference into the PICS alleged for any dependent patent claims that depend on such independent claims, as if fully set forth therein.

[7] The Accused Instrumentalities and associated exhibits discussed and/or cited for any claim herein are representative in all material respects of all other accused instrumentalities identified for that claim (e.g., a specified device or service may be used as a representative example in these charts

15.	A system, comprising[8]:	Microsoft uses, distributes, sells, and/or offers to sell a Universal Passwordless Architecture ("UPA") which includes: (a) the Microsoft Authenticator app (PROX_MSFT_003067, Plan a passwordless authentication deployment in Azure Active Directory – Microsoft Entra | Microsoft Learn) or Windows Hello,[9] (b) Microsoft Compatible FIDO2 Security Keys (PROX_MSFT_002844, Become a Microsoft-Compatible FIDO2 Security Key Vendor for sign-in to Azure AD - Microsoft Entra | Microsoft Learn) and/or Windows 10/11 which includes the Cloud Authentication Provider (CloudAP) and Web Account Manager (WAM), (c) Edge and Chrome browser support (PROX_MSFT_002819, All about FIDO2, CTAP2 and WebAuthn - Microsoft Community Hub), and (d) Microsoft Identity (a.k.a., Azure Active Directory and Microsoft Entra) (PROX_MSFT_003066, Passwordless is here and at scale - Microsoft Community Hub), which together infringe, literally or under the doctrine of equivalents, the claimed limitations as presented below. Microsoft, through its own actions, and the action of its partners, customers, and/or account holders, which Microsoft directs and controls, has created and sells the use of the UPA, including certifying devices and distributing servers and software supporting and integrating into the UPA architecture. Servers integrating with and supporting the architecture include Microsoft Identity servers providing FIDO and/or OAuth services. Devices integrating with and support the architecture includes Microsoft Compatible FIDO2 security keys. Software integrating with and supporting the architecture includes the Microsoft Edge web browser, support for the Chrome web browser, and the Microsoft Authenticator app and/or Windows Hello.
	a biometric key stores biometric data of a user and a plurality of codes and other data values comprising a device ID code uniquely identifying the biometric key and a secret decryption value	a biometric key stores biometric data of user … in a tamper proof format written to a storage element on the biometric key that is unable to be subsequently altered Security keys enable account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub ("You can now delete your password from your Microsoft account—or set up a new account with no password—and sign in

since the other accused instrumentalities have immaterial differences in their hardware and/or software configuration, the cited references are believed to be illustrative of all such accused devices).

[8] Plaintiff's inclusion of any claim preamble in this claim chart should not be interpreted as an admission that the preamble is limiting. Plaintiff reserves the right to take the position that the claim preambles are limiting or not limiting on a claim-by-claim basis.

[9] Windows Hello is an alternative to the Authenticator app detailed herein. Technical details regarding Windows Hello are described above with respect to claim 1 and, although systems including Microsoft Hello may infringe as reflected above (and for the same reasons), these contentions focus on the Authenticator app implementation.

	in a tamper proof format written to a storage element on the biometric key that is unable to be subsequently altered, and if scan data can be verified as being from the user by comparing the scan data to the biometric data, wirelessly sending, one or more codes from the plurality of codes and other data values wherein the one or more codes and the other data values include the device ID code, and the biometric data is selected from a group consisting of a palm print, a retinal scan, an iris scan, a hand geometry, a facial recognition, a signature recognition and a voice recognition; and	using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys."). To be used with Microsoft's Identity Platform, a security key needs to have FIDO2 certification. PROX_MSFT_002844, Become a Microsoft-Compatible FIDO2 Security Key Vendor for sign-in to Azure AD - Microsoft Entra | Microsoft Learn ("First, your authenticator needs to have a FIDO2 certification. We aren't able to work with providers who don't have a FIDO2 certification... Microsoft adds your FIDO2 Security Key on Azure Active Directory backend and to our list of approved FIDO2 vendors."). Accordingly, all authenticators within the system have met the standards set by Microsoft and FIDO. FIDO's CTAP protocol for authenticators includes a command for biometric enrollments. *See* PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.7. Accordingly, many of Microsoft's "current partners" provide authenticators storing biometrics. *See* PROX_MSFT_002844, Become a Microsoft-Compatible FIDO2 Security Key Vendor for sign-in to Azure AD - Microsoft Entra | Microsoft Learn (Providing a table of current partners indicating which provide "Biometric".). While the FIDO CTAP specification contains commands to enroll biometrics, it lacks commands to edit registered biometric enrollments. Such can be seen from the table below from the FIDO Alliance Client to Authenticator Protocol (CTAP). Missing from the table is any command to edit or alter a biometric enrollment. The list of sub commands for fingerprint(0x01) modality is: **subCommand Name** / **subCommand Number** enrollBegin — 0x01 enrollCaptureNextSample — 0x02 cancelCurrentEnrollment — 0x03 enumerateEnrollments — 0x04 setFriendlyName — 0x05 removeEnrollment — 0x06 getFingerprintSensorInfo — 0x07 PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.7. A biometric enrollment, accordingly, can only be created and deleted but not edited or altered once created. Furthermore, user verification is required by the FIDO CTAP specification. The first step the protocol defines for enrolling or deleting a fingerprint requires getting a user verification token with bio-enrollment permission. PROX_MSFT_002849, Client to Authenticator Protocol

		(CTAP) (fidoalliance.org), §§ 6.7.4 and 6.7.7 (“Platform gets pinUvAuthToken from the authenticator with the be permission.”). Subsequently, the enrollment or deletion procedures both verify the token and permission, aborting the procedures if verification fails or the bio-enrollment permission is not present. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), §§ 6.7.4 (“Authenticator calls verify(pinUvAuthToken, fingerprint (0x01) || enrollCaptureNextSample (0x02) || subCommandParams, pinUvAuthParam) If the verification fails, return CTAP2_ERR_PIN_AUTH_INVALID. Authenticator verifies that the pinUvAuthToken has be permission, if not, it returns CTAP2_ERR_PIN_AUTH_INVALID.”) and 6.7.7 (“Authenticator calls verify(pinUvAuthToken, fingerprint (0x01) || removeEnrollment (0x06) || subCommandParams, pinUvAuthParam) If the verification fails, return CTAP2_ERR_PIN_AUTH_INVALID. Authenticator verifies that the token has be permission, if not, it returns CTAP2_ERR_PIN_AUTH_INVALID."). As the biometric enrollments cannot be altered once created and cannot be created or deleted without user consent, FIDO compliant authenticators store biometric data of users in tamper proof format tht is unable to be subsequently altered. the biometric data is selected from a group consisting of a palm print, a retinal scan, an iris scan, a hand geometry, a facial recognition, a signature recognition and a voice recognition A subset of a palm print is a fingerprint, as a palm print includes at least one fingerprint. The built-in verification method present on FIDO compliant authenticators, like the authenticators Microsoft approves for use with the UPA, includes the use of biometrics, like fingerprints. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 5 (Defining “built-in User Verification method” as “The authenticator supports a built-in on-device user verification method like fingerprint…”). a biometric key stores … a plurality of codes and other data values comprising a device ID code uniquely identifying the biometric key … … in a tamper proof format written to a storage element on the biometric key that is unable to be subsequently altered Security keys enable account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub (“You can now delete your password from your Microsoft account—or set up a new account with no password—and sign-in using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys.”). To be used with Microsoft’s Identity Platform, a security key needs to have FIDO2 certification. PROX_MSFT_002844, Become a Microsoft-Compatible FIDO2 Security Key Vendor for sign-in to Azure AD -

		Microsoft Entra | Microsoft Learn ("First, your authenticator needs to have a FIDO2 certification. We aren't able to work with providers who don't have a FIDO2 certification... Microsoft adds your FIDO2 Security Key on Azure Active Directory backend and to our list of approved FIDO2 vendors."). Accordingly, all authenticators within the system have met the standards set by Microsoft and FIDO. The FIDO CTAP specification, with which authenticators in the system must comply, incorporates the WebAuthn Specification. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."). Under the WebAuthn specification, "compliant authenticators protect public key credentials." PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1. A public key credential refers to a public key credential source, which includes a credential ID. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining "public key credential" and "public key credential source".). The credential ID uniquely identifies its public key credential source. *See* Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a credential ID as "A probabilistically-unique byte sequence identifying a public key credential source and its authentication assertions."). In addition to the credential ID, each public key credential source contains a "credential private key". Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4, (Defining "public key credential source" as data structure including the credential private key and the credential ID.). "The credential private key is bound to a particular authenticator" and part of an asymmetric key pair containing a public key returned to a relying party. Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a "credential key pair".); and Passwordless authentication with Azure Active Directory - Microsoft Entra | Microsoft Learn ("Microsoft Authenticator uses key-based authentication to enable a user credential that is tied to a device, where the device uses a PIN or biometric."). Accordingly, a credential ID uniquely identifies a private/public key pair. While some identity providers, such as Google, may permit private keys to be shared across devices, Microsoft does not. PROX_MSFT_003062, Passkey support on Android and Chrome | Authentication | Google Developers ("Chrome on Windows stores passkeys in Windows Hello, which doesn't sync them to other devices as of January 2023."). Thus, every public/private key pair will be unique to a specific security key. Uniquely identifying device-specific public/private key pairs, the credential ID necessarily uniquely identifies the device, making it a device ID. The credential ID is used to reference the corresponding public key during authentication ceremonies. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key

		Credentials - Level 2 (w3.org), § 7.2 ("Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key."). It is, therefore, necessary that the credential ID be unique within the system, otherwise, the wrong key will be located, and authentication will fail. Accordingly, the WebAuthn specification requires deleting older registrations with an identical credential ID or failing the registration of the new credential. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.1 ("Check that the credentialId is not yet registered to any other user. If registration is requested for a credential that is already registered to a different user, the Relying Party SHOULD fail this registration ceremony, or it MAY decide to accept the registration, e.g. while deleting the older registration."). In addition to uniquely identifying the biometric key, the credential ID is in a tamper proof format, unable to be subsequently altered. A device ID cannot be created without the consent of the relying party (i.e., Microsoft). As previously noted, a device ID (i.e., credential ID) is an element of a Public Key Credential. *See* PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a Public Key Credential as referencing a "public key credential source", which in turn is defined as including a credential ID). Being an element of the credential, each time a credential is created a new credential ID will be created. The process of creating a new credential is called a "Registration Ceremony" and is initiated by the relying party, i.e. Microsoft. *See* PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 (Defining a Registration Ceremony as "the ceremony where a user, a Relying Party, and the user's client (containing at least one authenticator) work in concert to create a public key credential and associate it with the user's Relying Party account" and stating such "is initiated by the Relying Party calling navigator.credentials.create() with a publicKey argument."). As the Relying Party initiates the creation of a new credential, including the associated credential ID, a credential ID will never be created without Microsoft's consent. In addition to not being able to be created without consent, credential IDs cannot be modified once created. The FIDO CTAP standards lack the functionality to modify a credential ID. Within FIDO Authenticators, credentials are managed using the authenticatorCredentialManagement command. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.8. As shown in the table below, the command does not include an option to modify a credential ID.

The list of sub commands for credential management is:

subCommand Name	subCommand Number
getCredsMetadata	0x01
enumerateRPsBegin	0x02
enumerateRPsGetNextRP	0x03
enumerateCredentialsBegin	0x04
enumerateCredentialsGetNextCredential	0x05
deleteCredential	0x06
updateUserInformation	0x07

The only thing that may be edited is user information. User information, however, corresponds to the WebAuthn value userEntity, as indicated in the table below.

[WebAuthn] `authenticatorMakeCredential` operation	CTAP `authenticatorMakeCredential` operation
hash	clientDataHash
rpEntity	rp
userEntity	user
requireResidentKey	options.rk
requireUserPresence	options.up Note: [WebAuthn-2] defines `requireUserPresence` as a constant Boolean value `true`. options.up is required to be absent for backwards comparability with CTAP2.0.
requireUserVerification	options.uv or pinUvAuthParam
credTypesAndPubKeyAlgs	pubKeyCredParams
excludeCredentialDescriptorList	excludeList
extensions	extensions

PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), §6.1.

The WebAuthn standard, in turn, defines userEntity as containing data about the user account for which the credential is created, and having the required fields of name, displayName and id. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 5.4 ("This member contains data about the user account for which the Relying Party is requesting attestation. Its value's name, displayName and id members are REQUIRED."). User Information, accordingly, does not include the credential ID. Updating user information, therefore, will not change the credential ID.

Furthermore, altering the credential ID causes verification to fail. "[A]ll that would happen if an authenticator returns the wrong credential ID, or if an attacker intercepts and manipulates the credential ID, is that the WebAuthn Relying Party would not look up the correct credential public key with which to verify the returned signed authenticator data (a.k.a., assertion), and thus the interaction would end in an error." PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 13.1.

As altering the device ID (i.e., Credential ID) causes authentication to fail, the authenticator (i.e., biometric key) lacks commands to alter the credential ID, and the credential ID cannot be created without the consent of Microsoft, the credential ID is necessarily in a tamper proof format unable to be altered.

a biometric key stores … a secret decryption value in a tamper proof format written to a storage element on the biometric key that is unable to be subsequently altered

Security keys enable account holders to delete passwords from their account in favor of a more secure and convenient authentication method. PROX_MSFT_003036, Introducing password removal for Microsoft Accounts - Microsoft Community Hub ("You can now delete your password from your Microsoft account—or set up a new account with no password—and sign-in using other more secure and convenient authentication methods such as the Microsoft Authenticator app, Windows Hello, or physical security keys."). To be used with Microsoft's Identity Platform, a security key needs to have FIDO2 certification. PROX_MSFT_002844, Become a Microsoft-Compatible FIDO2 Security Key Vendor for sign-in to Azure AD - Microsoft Entra | Microsoft Learn ("First, your authenticator needs to have a FIDO2 certification. We aren't able to work with providers who don't have a FIDO2 certification... Microsoft adds your FIDO2 Security Key on Azure Active Directory backend and to our list of approved FIDO2 vendors."). Accordingly, all authenticators within the system have met the standards set by Microsoft and FIDO.

The FIDO CTAP specification, to which biometric keys in the system must comply, "is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification." PROX_MSFT_002849. Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1. The WebAuthn specification is based on asymmetric cryptography such that a private cryptographic key is stored by the authenticator and its corresponding public cryptographic key is stored by the relying party (i.e. Microsoft Identity Platform). PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 ("At registration time, the authenticator creates an asymmetric key pair, and stores its private key portion and information from the Relying Party into a public key credential source. The public key portion is returned to the Relying Party, who then stores it in conjunction with the present user's account."). Private keys are notoriously known as secret decryption values.

By being part of a pair split between the authenticator and the relying party, the private key is in a tamper proof format unable to be subsequently altered. During authentication, the relying party looks up the appropriate key based on the credential ID (i.e., device ID) supplied by the

authenticator and verifies a signature generated using the authenticator's private key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash.") If the authenticator's private key was altered, it would generate a signature that could not be verified with the public key held by the relying party. Such would cause the relying party to "fail the authentication ceremony." PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("22. If all the above steps are successful, continue with the authentication ceremony as appropriate. Otherwise, fail the authentication ceremony."). Accordingly, by being part of a pair divided between the relying party (i.e., Microsoft Identity Platform) and the authenticator, the private key is in a tamper proof format unable to be subsequently altered.

In the alternative, to the extent the above is considered to be a different security implementation or if limited alterations are possible in limited circumstances and given high levels of security permissions, clearance, or hardware-level permission attribution, the "tamper proof format . . . that is unable to be subsequently altered," limitation is met under the doctrine of equivalents on the grounds that the tamper proofing and limitations on alteration required for securit keys must prevent nearly all alteration by nearly all programs that have not been explicitly approved and, at least, prevent alteration and requires a strict level of physical and/or virtual permission to be altered. Moreover, this limitation could be met under the doctrine of equivalents (if not directly infringed) if alterations to the secrety keys break the system's connection with the stored Credential ID and render the Credential ID unusable as this is the same function, way, and result as preventing alterations entirely in the first place.

if scan data can be verified as being from the user by comparing the scan data to the biometric data,

When the FIDO2 security key (i.e., biometric key) receives an authentication request in the form of an authenticatorGetAssertion request with the user verification option present and set as true, the authenticator will attempt user verification, returning an error if user verification is not successful. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 ("2. If the "uv" option is present and set to true (implying the pinUvAuthParam parameter is not present, and that the authenticator supports an enabled built-in user verification method, see Step 4): 1. Let internalRetry be true. 2. Let uvState be the result of calling performBuiltInUv(internalRetry) 3. If uvState is error: 1. If the error reason is a user action

timeout, then return CTAP2_ERR_USER_ACTION_TIMEOUT. 2. If the ClientPin option ID is true and the noMcGaPermissionsWithClientPin option ID is absent or false, end the operation by returning CTAP2_ERR_PUAT_REQUIRED. 3. If the uvRetries counter is <= 0, return CTAP2_ERR_PIN_BLOCKED. 4. Otherwise, end the operation by returning CTAP2_ERR_OPERATION_DENIED. 4. If uvState is success: 1. Set the "uv" bit to true in the response."). As such, only an error will be returned if scan data cannot be verified. Otherwise, the process of generating a response including the credential ID will continue.

wirelessly sending, one or more codes from the plurality of codes and other data values wherein the one or more codes and the other data values include the device ID code

FIDO's CTAP protocol is intended to be used with "roaming authenticators." PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1 ("This protocol is intended to be used in scenarios where a user interacts with a Relying Party (a website or native app) on some platform (e.g., a PC) which prompts the user to interact with a roaming authenticator (e.g., a smartphone)."). Roaming authenticators are removable from, and can 'roam' between, client devices. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 6.2.1 ("A roaming authenticator is attached using cross-platform transports, called cross-platform attachment. Authenticators of this class are removable from, and can "roam" between, client devices."). FIDO's CTAP protocol permits requests be conveyed to and responses received from a roaming authenticator via BLE (i.e., Bluetooth) and NFC – both of which are wireless means of sending and receiving data. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 3 ("Requests and responses are conveyed to roaming authenticators over specific transports (e.g., USB, NFC, Bluetooth).").

During an authentication ceremony, an authenticator receives an "authenticatorGetAssertion" request to provide cryptographic proof of user authentication. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2 (Defining authenticatorGetAssertion as the method "used by a host to request cryptographic proof of user authentication as well as user consent to a given transaction, using a previously generated credential that is bound to the authenticator and relying party identifier."). When successfully invoked, the authenticatorGetAssertion causes the authenticator to return a response including a "PublicKeyCredentialDescriptor structure containing the credential identifier whose private key was used to generate the assertion." PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2. As noted above, the credential ID is a device ID uniquely identifying the authenticator. The authenticator will thus wirelessly send via BLE or NFC the device ID.

	an authentication unit receives the plurality of codes and the other data values and send the plurality of codes and the other data values to agent for authentication to determine whether the one or more codes and the other data values are legitimate, wherein the agent is a third-party trusted authority possessing a list of device ID codes uniquely identifying legitimate integrated devices, and responsive to the device ID code being authenticated, the authentication unit receiving an access message from the agent allowing the user to access an application, wherein the application is selected from a group consisting of a casino machine, a keyless lock, a garage door opener, an ATM machine, a hard drive, computer software, a web site and a file.	an authentication unit receives the plurality of codes and other data values Browsers, such as Microsoft Edge and Google Chrome, operating on Windows 11 form an authentication unit that receives the one or more codes and other data values. PROX_MSFT_003282, Passwordless login with passkeys | Authentication | Google Developers ("When a user wants to sign in to a service that uses passkeys, their browser or operating system will help them select and use the right passkey."); and PROX_MSFT_002819, All about FIDO2, CTAP2 and WebAuthn - Microsoft Community Hub ("CTAP2 and WebAuthn define an abstraction layer that creates an ecosystem for strongly authenticated credentials. Any interoperable client (such as a native app or browser) running on a given 'client device' can use a standardized method to interact with any interoperable authenticator – which could mean a platform authenticator that is built into the client device or a roaming authenticator that is connected to the client device through USB, BLE, or NFC."). Both Microsoft Edge and Google Chrome web browsers support the use of FIDO's CTAP protocol. PROX_MSFT_002819, All about FIDO2, CTAP2 and WebAuthn - Microsoft Community Hub ("Microsoft Edge plays the part of a WebAuthn Client. Edge can handle the UI for the above listed WebAuthn/CTAP2 features, and also supports the AppID extension. Edge is capable of interacting with both CTAP1 and CTAP2 authenticators, which means that it can facilitate the creation and use of both U2F and FIDO2 credentials…"); and PROX_MSFT_003062, Passkey support on Android and Chrome | Authentication | Google Developers ("Chrome on all desktop platforms supports using passkeys from mobile devices."). Windows supports the use of BLE and NFC roaming authenticators with Edge and Chrome web browsers. *See* PROX_MSFT_002847, Browser support of FIDO2 passwordless authentication - Microsoft Entra | Microsoft Learn (Presenting a table indicating Windows supports NFC and BLE passkeys with Microsoft Accounts "created by consumers for services such as Xbox, Skype, or Outlook.com" via the Chrome and Edge web browsers.). Regardless of the web browser, Windows provides WebAuthn APIs, enabling interactions with authenticators to take place. PROX_MSFT_002819, All about FIDO2, CTAP2 and WebAuthn - Microsoft Community Hub ("Windows 10 plays the part of the platform, hosting Win32 Platform WebAuthn APIs that enable clients to interact with Windows Hello in order for users to be prompted and interactions with authenticators to take place.") As such, Windows will connect with a roaming authenticator over BLE or NFC to receive the credential necessary from the roaming authenticator, enabling the user to sign into a website or application accessed using either the Edge or Chrome web browser. an authentication unit … send the plurality of codes and the other data values to agent for authentication

The responsibility for sending responses received from an authenticator via BLE or NFC falls upon the WebAuthn Client. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 4 ("A WebAuthn Client is an intermediary entity typically implemented in the user agent (in whole, or in part). Conceptually, it underlies the Web Authentication API and embodies the implementation of the [[Create]](origin, options, sameOriginWithAncestors) and [[DiscoverFromExternalSource]](origin, options, sameOriginWithAncestors) internal methods. It is responsible for both marshalling the inputs for the underlying authenticator operations, and for returning the results of the latter operations to the Web Authentication API's callers."). Microsoft identifies its Edge Browser as the WebAuthn client. PROX_MSFT_002819, All about FIDO2, CTAP2 and WebAuthn - Microsoft Community Hub ("Microsoft Edge plays the part of a WebAuthn Client. Edge can handle the UI for the above listed WebAuthn/CTAP2 features, and also supports the AppID extension. Edge is capable of interacting with both CTAP1 and CTAP2 authenticators, which means that it can facilitate the creation and use of both U2F and FIDO2 credentials…"). As such, Microsoft's Edge Browser will forward the credential received from a roaming authenticator over BLE or NFC to the proper authority.

Google identifies its Chrome browser as a WebAuthn client by noting it supports the use of passkeys from mobile devices, including permitting the use of an Android device as a roaming authenticator on Windows. PROX_MSFT_003062, Passkey support on Android and Chrome | Authentication | Google Developers ("Chrome on all desktop platforms supports using passkeys from mobile devices."); and PROX_MSFT_003062, Passkey support on Android and Chrome | Authentication | Google Developers (Providing a table indicating "Can sign in with phone" is supported on Chrome running on the macOS, iOS, and Windows.). As such, Chrome will forward the credential received from the roaming authenticator on a device to the proper authority.

to agent for authentication to determine whether the one or more codes and the other data values are legitimate

Authentication is provided by a FIDO server operated by Microsoft. FIDO incorporates the WebAuthn Protocol. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1.1 ("Along with the Web Authentication API itself, this specification defines a request-response cryptographic protocol—the WebAuthn/FIDO2 protocol—between a WebAuthn Relying Party server and an authenticator, where the Relying Party's request consists of a challenge and other

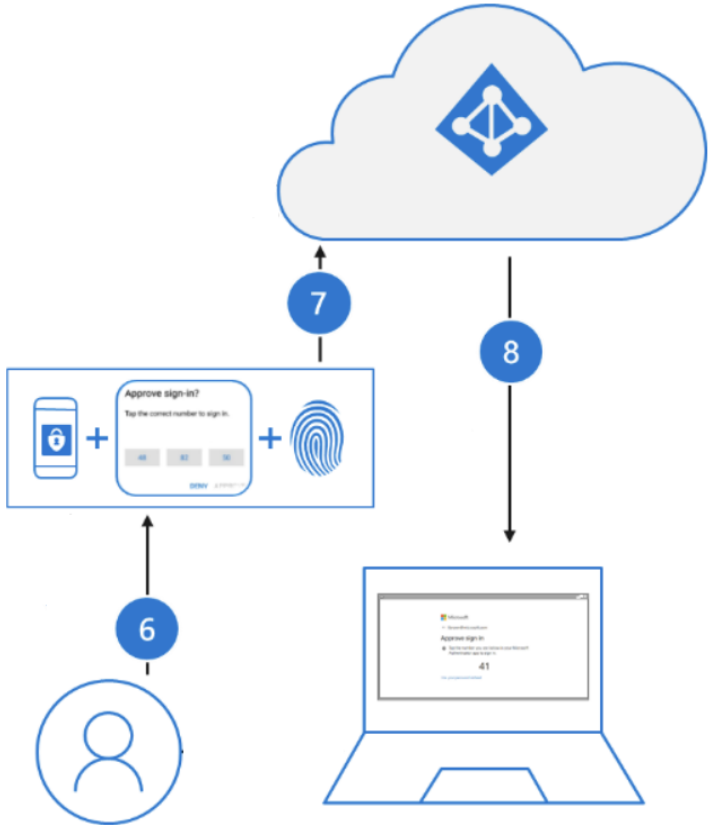
input data supplied by the Relying Party and sent to the authenticator.") Accordingly, the FIDO server challenge sent to the roaming authenticator will be a WebAuthn authenticatorGetAssertion request. The response to the request received will contain a signature signed by the authenticator's private key and a credential ID. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 (Detailing the response to authenticationGetAssertion as containing "PublicKeyCredentialDescriptor structure containing the credential identifier whose private key was used to generate the assertion" and "an assertion signature".); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 6 (Defining an "assertion signature" as "an assertion signature asserts that the authenticator possessing a particular credential private key has established, to the best of its ability, that the user requesting this transaction is the same user who consented to creating that particular public key credential."). Upon receiving the response, the WebAuthn/FIDO server will use the credential ID to locate the appropriate public key to verify the signature generated with the private key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash.") Accordingly, a FIDO/WebAuthn server operated by Microsoft Identity will authenticate a device ID (i.e., credential ID) to determine if it is legitimate by verifying of a signature generated with a corresponding private key.

wherein the agent is a third-party trusted authority

The Microsoft Identity Platform utilizes OpenID Connect and Microsoft encourages developers to gain an understanding of the protocol and concepts to add authentication to applications. PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("Knowing about OAuth or OpenID Connect (OIDC) at the protocol level isn't required to use the Microsoft identity platform. However, you'll encounter protocol terms and concepts as you use the identity platform to add authentication to your apps.").

Within the protocol, Microsoft identifies its identity platform as an "authorization server" managing "trust relationships" and issuing security tokens applications and APIs use for granting access (i.e. authorization) and authentication. PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("The identity platform is the authorization server. Also called an identity provider or IdP, it securely handles the end-users information, their access, and the trust relationships between the parties in

the auth flow. The authorization server issues the security tokens your apps and APIs use for granting, denying, or revoking access to resources (authorization) after the user has signed in (authenticated)."). One of the tokens issued by Microsoft's Identity Platform is an ID Token used when signing in users. PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("ID tokens are issued by the authorization server to the client application. Clients use ID tokens when signing in users and to get basic information about them."). Accordingly, Microsoft describes its Identity Platform as a third party that operates a trusted authority for authentication.



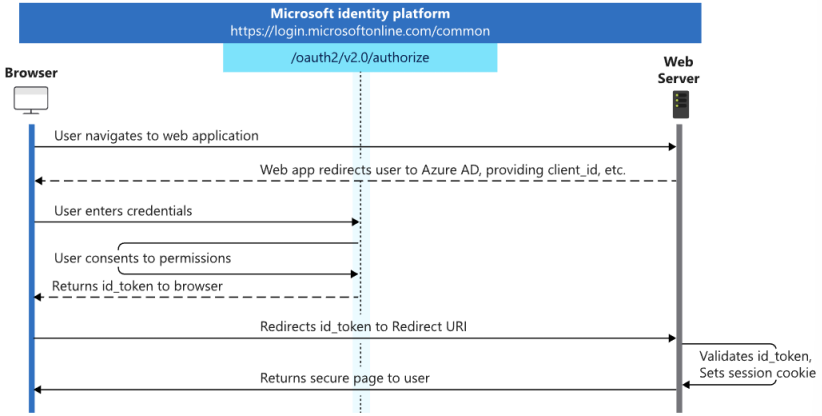
6. The user completes the challenge by entering their biometric or PIN to unlock private key.
7. The nonce is signed with the private key and sent back to Azure AD.
8. Azure AD performs public/private key validation and returns a token.

		See PROX_MSFT_002830, Azure Active Directory passwordless sign-in - Microsoft Entra | Microsoft Learn, https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless. Additionally, as shown in the above excerpt from Microsoft's description of its UPA, Azure Active Directory is an agent that is a third party trusted authority. Azure AD is being renamed Microsoft Entra ID. PROX_MSFT_002825, https://azure.microsoft.com/en-us/products/active-directory. Azure AD forms the core of the Microsoft Identity Platform. Accordingly, in the context of the above and cited documentation herein, Identity Platform, Azure AD, and/or Entra ID are all referencing the same third party trusted authority system. possessing a list of device ID codes uniquely identifying legitimate integrated devices As noted supra, the FIDO server within the Microsoft Identity Platform uses a credential ID operating as a device ID to verify a signature generated with a private key bound to the identified device (i.e., authenticator). Such action is made possible by registering the credentials with an account and associating the account with the credential ID and credential public key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.1. (Detailing the registration ceremony as including "register the new credential with the account that was denoted in options.user" and "Associate the user's account with the credentialId and credentialPublicKey in authData.attestedCredentialData".). Accordingly, the Microsoft Identity Platform maintains a listing of legitimate device IDs. responsive to the device ID code being authenticated, the authentication unit receiving an access message from the agent allowing the user to access an application Authentication is provided by a FIDO server operated by Microsoft in conjunction with OpenID Connect Authentication Server. While OpenID Connect is a federation protocol, it is compatible and complementary with FIDO. "The value of a FIDO authentication capability is amplified by a federated system, where the federation system extends the benefits of a FIDO authentication to applications and services without requiring FIDO to be directly integrated with those applications." PROX_MSFT_003006, Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf (fidoalliance.org), page 4. When a federated system, such as OpenID Connect, is combined with FIDO, the OpenID Provider (OP) sends a FIDO server challenge to an authenticator which is returned as the FIDO Authentication response. PROX_MSFT_003006, Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf (fidoalliance.org), page 10

		(Detailing the combined use FIDO and federation entails "3… The Authentication Authority determines that the FIDO authentication specified in the previous request is both relevant… 4. The Authentication Authority sends a FIDO Server challenge… 6. FIDO Client returns FIDO authentication response… 7. FIDO Server validates the FIDO response and reports same to Authentication Authority which looks up the user and then redirects the User Agent back to the Application Provider with an authentication assertion."). FIDO incorporates the WebAuthn Protocol. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 1.1 ("This specification is part of the FIDO2 project, which includes this specification and is related to the W3C [WebAuthn] specification."); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 1.1 ("Along with the Web Authentication API itself, this specification defines a request-response cryptographic protocol—the WebAuthn/FIDO2 protocol—between a WebAuthn Relying Party server and an authenticator, where the Relying Party's request consists of a challenge and other input data supplied by the Relying Party and sent to the authenticator.") Accordingly, the FIDO server challenge sent will be a WebAuthn authenticatorGetAssertion request. The response to the request received will contain a signature generated by the authenticator's private key and a credential ID. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 (Detailing the response to authenticationGetAssertion as containing "PublicKeyCredentialDescriptor structure containing the credential identifier whose private key was used to generate the assertion" and "an assertion signature".); and PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 6 (Defining an "assertion signature" as "an assertion signature asserts that the authenticator possessing a particular credential private key has established, to the best of its ability, that the user requesting this transaction is the same user who consented to creating that particular public key credential."). Upon receiving the response, the WebAuthn/FIDO server will use the credential ID to locate the appropriate public key to verify the signature generated with the private key. PROX_MSFT_003098, Web Authentication: An API for accessing Public Key Credentials - Level 2 (w3.org), § 7.2 ("7. Using credential.id (or credential.rawId, if base64url encoding is inappropriate for your use case), look up the corresponding credential public key and let credentialPublicKey be that credential public key… 20. Using credentialPublicKey, verify that sig is a valid signature over the binary concatenation of authData and hash.") Accordingly, a FIDO/WebAuthn server operated by Microsoft Identity Platform will authenticate a device ID (i.e., credential ID). After validating the device ID, the FIDO/WebAuthn portion of Identity Platform "redirects the user agent back to the Application Provider with an authentication assertion". PROX_MSFT_003006, Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf (fidoalliance.org), page 10.

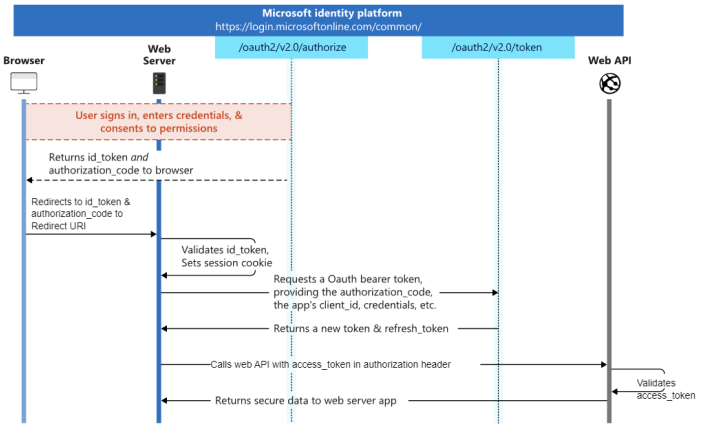
As the Microsoft Identity Platform utilizes OpenID Connect, the authentication assertion will be an ID Token. OpenID Connect defines an ID Token as "a security token that contains Claims about the Authentication of an End-User by an Authorization Server when using a Client, and potentially other requested Claims."

The ID Token received from Microsoft's Identity platform enables access to an application. Such is clearly indicated in the below flow chart of Microsoft's implementation of OpenID Connect. As can be seen, at the end of the process, the ID Token received is exchanged for access to an application.



PROX_MSFT_003047, [OpenID Connect (OIDC) on the Microsoft identity platform - Microsoft Entra | Microsoft Learn]

The ID Token is thus an access message provided by Microsoft operating as third party trusted authority allowing access to an application. Should the user also require an access token to access further features of the application, such as files containing stored data, the ID token can be exchanged for an access token utilizing the flow detailed below.

		 PROX_MSFT_003047, OpenID Connect (OIDC) on the Microsoft identity platform - Microsoft Entra | Microsoft Learn wherein the application is selected from a group consisting of a casino machine, a keyless lock, a garage door opener, an ATM machine, a hard drive, computer software, a web site and a file. As noted above, Microsoft utilizes Open ID Connect, which includes a client that Microsoft identifies as including web apps and web APIs, both of which represent software, or single page within the browser (i.e., website). PROX_MSFT_003042, OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform - Microsoft Entra | Microsoft Learn ("The client could be a web app running on a server, a single-page web app running in a user's web browser, or a web API that calls another web API. You'll often see the client referred to as client application, application, or app."). Accordingly, once the account holder has completed passwordless authentication via the Microsoft Authenticator App, they can access Office 365 applications and files, Xbox Live, synced credit cards, and other applications hosted by the Microsoft's resource servers (i.e., clients) and the resource servers of Microsoft Identity's customers.
16	The system of claim 15, wherein the biometric key receives an authentication request from the authentication unit, and, in response, requests a biometric scan from the user to generate the scan data.	When the FIDO2 security key (i.e., biometric key) receives an authentication request in the form of an authenticatorGetAssertion request with the user verification option present and set as true, the authenticator will attempt user verification, returning an error if user verification is not successful. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 ("2. If the "uv" option is present and set to true (implying the pinUvAuthParam parameter is not present, and that the authenticator supports an enabled built-in user verification method, see Step 4): 1. Let internalRetry be true. 2. Let uvState be the result of calling performBuiltInUv(internalRetry) 3. If uvState is error: 1. If the error reason is a user action timeout, then return CTAP2_ERR_USER_ACTION_TIMEOUT. 2. If

		the ClientPin option ID is true and the noMcGaPermissionsWithClientPin option ID is absent or false, end the operation by returning CTAP2_ERR_PUAT_REQUIRED. 3. If the uvRetries counter is <= 0, return CTAP2_ERR_PIN_BLOCKED. 4. Otherwise, end the operation by returning CTAP2_ERR_OPERATION_DENIED. 4. If uvState is success: 1. Set the "uv" bit to true in the response.”).
17	The system of claim 15, wherein if the biometric key cannot verify the scan data as being from the user, it does not send one or more codes and the other data values.	When the FIDO2 security key (i.e., biometric key) receives an authentication request in the form of an authenticatorGetAssertion request with the user verification option present and set as true, the authenticator will attempt user verification, returning an error if user verification is not successful. PROX_MSFT_002849, Client to Authenticator Protocol (CTAP) (fidoalliance.org), § 6.2.2 ("2. If the "uv" option is present and set to true (implying the pinUvAuthParam parameter is not present, and that the authenticator supports an enabled built-in user verification method, see Step 4): 1. Let internalRetry be true. 2. Let uvState be the result of calling performBuiltInUv(internalRetry) 3. If uvState is error: 1. If the error reason is a user action timeout, then return CTAP2_ERR_USER_ACTION_TIMEOUT. 2. If the ClientPin option ID is true and the noMcGaPermissionsWithClientPin option ID is absent or false, end the operation by returning CTAP2_ERR_PUAT_REQUIRED. 3. If the uvRetries counter is <= 0, return CTAP2_ERR_PIN_BLOCKED. 4. Otherwise, end the operation by returning CTAP2_ERR_OPERATION_DENIED. 4. If uvState is success: 1. Set the "uv" bit to true in the response.”). As such, only an error will be returned if scan data cannot be verified.